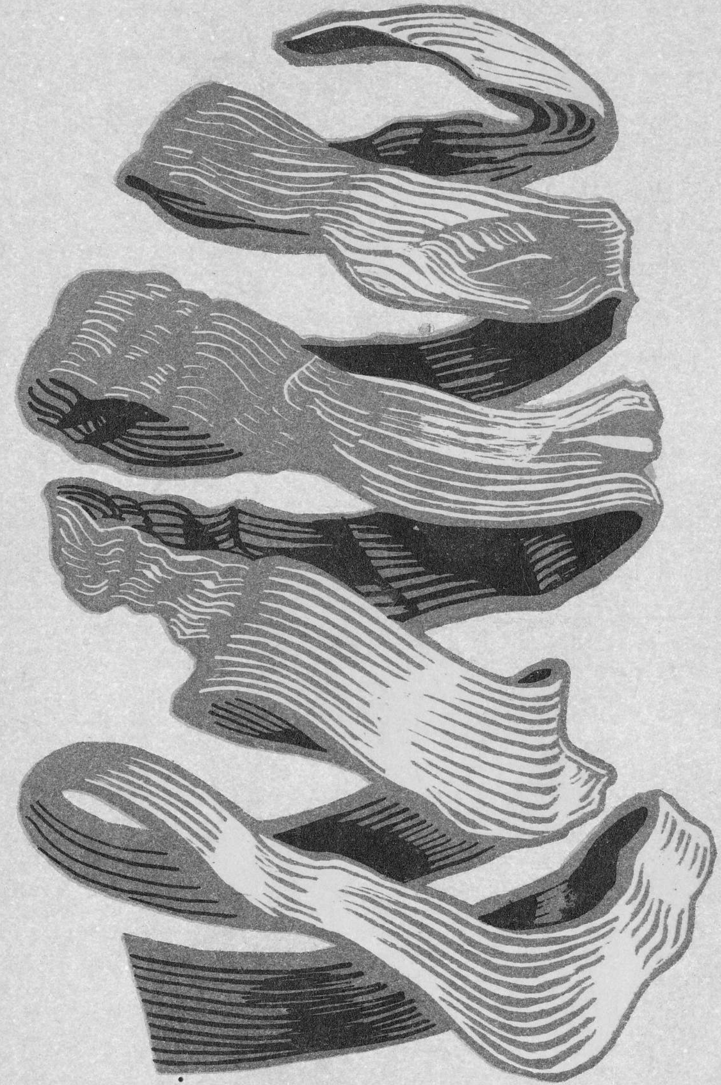
Dr. Ioan Andone



EDITURA MOLDOVA



inteligență artificială și sisteme expert în contabilitate

inteligență artificială și sisteme expert în contabilitate

Lucrarea de față își propune să abordeze într-o optică deosebit de actuală cele mai noi direcții de modernizare a contabilității — sistemele inteligente, în special sistemele expert. Ea se adresează tuturor celor interesați în efortul comun de deschidere a noilor orizonturi în știință și aplicațiile contabilității. Pentru aceasta vom aborda succesiv și intercorelat problematica inteligenței artificiale și sistemelor expert în contabilitate, cu demonstrarea folosirii acestor tehnologii de vîrf în produse informatice contabile inteligente.

Tuturor cititorilor le sîntem recunoscători pentru eventualele observații și sugestii de îmbunătățire.

Dr. Ioan ANDONE

moldova

Lei 500

ISBN 973-9032-86-9

"Educaţia este al doilea soare pentru cei care o au."

Heraclit

CUVÎNT ÎNAINTE

Sîntem martorii şi protagoniştii unei revoluţii ştiinţifice şi tehnice fără precedent, proces cu ample implicaţii în toate domeniile vieţii economice şi sociale, care suscită un interes deosebit din partea economiştilor cercetători în specialitatea contabilitate şi nu numai.

Privind epocile istorice observăm că progresul ştiinţifico-tehnic a reprezentat, în fond, un instrument în mîna omului, pentru că a pus toate ramurile ştiinţei în faţa unor noi cerinţe.

În ceea ce priveşte ştiinţa contabilităţii, în sensul cel mai restrîns, ea beneficiază în prezent de principiile teoriei sistemelor, principiile ciberneticii şi metodele informaticii pentru constituirea de sisteme integrate şi sisteme expert de mare performanţă, în necontenita îmbogăţire şi adîncire a cîmpului cunoaşterii în materia contabilă, datorită posibilităţii transferului de inteligenţă de la om către produsele informatice cu specific contabil. Acest lucru înseamnă în primul rînd perfecţionarea teoriei prin dezvoltarea conceptelor şi principiilor, urmată de procesul unitar şi dialectic de asimilare creatoare a realizărilor conceptuale din ştiinţele de graniţă, în vederea constituirii celui mai perfecţionat

și eficient sistem de informare și control asupra activității agenților economici, pentru sporirea contribuției contabilității la fundamentarea științifică a deciziilor la toate nivelurile.

Diversitatea, cantitatea și costul tot mai ridicat al informației contabile sînt praguri peste care nu se poate trece decît cu forța mijloacelor tehnice moderne și cu o concepție unitară integratoare.

Știința contabilității, cu deosebire în ultimele decenii ale secolului nostru, a înregistrat puternice momente de deschidere sistemică, pentru a avea o mai bună cuprindere teoretică și aplicativă a întregii arii de cercetare și pentru a putea răspunde la cerințele de modernizare.

În acest context trebuie să remarcăm o idee de esență - modificările calitative în domeniul tehnicii contabile s-au produs întotdeauna în sensul unei complexități crescînde iar epoca cibernetizării și informatizării ține latura aplicativă a contabilității sub incidența fertilă de perspectivă. În consecință, modul de gîndire integrator-dinamic specific sfîrșitului de secol XX se aplică și în contabilitate, oferindu-ne o posibilitate mai bună de a sesiza valabilitatea principiilor sale axiale aplicate la cunoașterea realității investigate.

De pe această poziție trebuie abordate conceptele, principiile și modelele contabilității, dovedind respectul cuvenit pentru efortul înaintașilor, care au cristalizat doctrinele și fondul principal de cunoștințe științifice, în mod sistematic, dîndu-ne posibilitatea să investigăm astăzi noile frontiere ale acestei discipline.

Cadrul sistematic în care putem înțelege locul de acțiune al contabilității este întreprinderea modernă concepută ca sistem (societatea comercială, regia autonomă, etc.). Dacă aceasta ar fi o motivație a demersului științific pe care îl facem, altă cauză cu siguranță mai importantă este aceea că știința contabilității se înscrie în rîndul factorilor de progres, ea beneficiază de un statut privelegiat în rîndul disciplinelor de economie, informațiile ei pline de realism și previziune comunicate factorilor decizionali și altor agenți economici interesați, permit judecăți de valoare, slujind promovarea amplă a progresului material și moral al societății.

Contabilitatea este legiferată și se aplică în toate sistemele economico-sociale (întreprinderi, alte organizații economice, familia, grupuri și organizații sociale, economia națională și relațiile economice internaționale), care realizează activități cu finalitate economică și servicii utile societății.

În cadrul acestor sisteme, contabilitatea observă ansamblul mișcărilor

de valori exprimate în bani precum și raporturile economico-juridice care provoacă decontări bănești, starea, mișcarea și transformarea mijloacelor economice și resursele în ordinea lor de formare și după destinație în procesul de reproducție. Esențial în legătură cu contabilitatea este să observăm că tot ea oferă instrumentele intelectuale necesare stabilirii rezultatelor unei activități economice și măsurarea rentabilității, fapt care accentuează importanța informațiilor sale ca nucleu central în sistemele informaționale moderne.

Astăzi sînt mai stringente ca oricînd deschiderile acestei științe pentru estimarea valorii bunurilor materiale consumate, a resurselor deteriorate, a serviciilor de tot felul care trebuie incluse în avuția națională. Știința contabilității oferă informații pertinente asupra conservării și dezvoltării avuției naționale în condițiile ocrotirii mediului ambiant și creșterii bunăstării și armoniei sociale.

Pentru contabilitate, sistemele informatice și sistemele expert - ca nuclee ale sistemelor informaționale modernizate - prezintă deosebit interes datorită performanțelor obținute în prelucrarea informațiilor (date și cunoștințe) în special în planul calității, corectitudinii și operativității, realizîndu-i mai bine obiectivele. Practica actuală a concretizat sisteme informatice contabile care integrează sisteme expert capabile să amplifice acțiunile de cunoaștere a rezultatelor pozitive și să semnalizeze neajunsurile de ordin pecuniar și mai mult, să intervină chiar cu modificări în timp real a indicatorilor economici ori să introducă noi concepte și principii de cunoaștere contabilă.

În prezent informatizarea contabilității asigură manifestarea plenară a principiilor și metodei contabilității, a sistematizării, prelucrării și raportării informațiilor pe calea bilanțului și a celorlalte rapoarte și situații financiar-contabile. Generalizarea folosirii teleinformaticii și posibilitatea exploatării în regim interactiv a bazelor de date au deschis noi orizonturi pregătirii informației operative privind costurile, încadrarea în bugetele de venituri și cheltuieli și punerea în valoare a funcției previzionale a contabilității.

Contabilitatea ca știință a depășit de mult faza acumulărilor, a intrat în cea a saltului calitativ. Ea s-a afirmat ca o cvasiștiință economică și socială în toate drepturile sale, fiind prezentă în orice cultură. În rîndul disciplinelor științifice și a tehnologiilor lor proprii, contabilitatea este și va rămîne o știință modernă prin arsenalul ei de concepte, metoda ei rațională și tehnica specifică mereu perfecționată, ocupîndu-se de probleme prioritare și răspunzînd dezvoltării economiei pe principii de eficiență.

* *
*

Lucrarea de față își propune să abordeze într-o optică deosebit de actuală cele mai noi direcții de modernizare a contabilității - sistemele inteligente, în special sistemele expert. Ea se adresează tuturor celor interesați în efortul comun de deschidere a noilor orizonturi în știință și aplicațiile contabilității. Pentru aceasta vom aborda succesiv și interco-relat problematica inteligenței artificiale și sistemelor expert în contabili-tate, cu demonstrarea folosirii acestor tehnologii de vîrf în produse informatice contabile inteligente.

Tuturor cititorilor le sîntem recunoscători pentru eventualele obser-vații și sugestii de îmbunătățire.

Iași, 22 mai 1991

Dr. Ioan Andone

Capitolul I

ABORDAREA SISTEMICĂ ȘI INFORMATIZAREA CONTABILITĂȚII

1.1. Sistem informatic. Concept și arhitectură funcțională

Din punct de vedere funcțional sistemul informatic are mai multe definiții, în funcție de scop, elementele din structură și relațiile dintre elemente.

Cea mai generală definiție abordează sistemul informatic ca pe un sistem de colectare, memorare, prelucrare și distribuire a informațiilor care utilizează calculatoarele electronice.

Scopul principal al sistemului informatic este de a servi cerințele informaționale ale managerilor de la diferite niveluri decizionale și de a reduce la minimum intervenția și efortul uman în desfășurarea unor procese - de concepție, de producție, de gestiune, de decizie etc.

Fiecărui sistem informatic i se asociază un sistem de prelucrare a datelor, în care datele se prezintă pe diferiți suporți de memorare iar procesele de prelucrare sînt concretizate în proceduri (automate și/sau manuale) executate de către diferite echipamente de tehnică de calcul și personal specializat. Mai nou sistemele informatice integrează și sisteme expert, programe specializate în prelucrarea cunoștințelor.

Din acest punct de vedere, sistemul informatic poate fi considerat ca un ansamblu de oameni, informații, date și proceduri de prelucrare, echipamente de calcul și comunicații.

Există opinii [Turbuț, 68] potrivit cărora sistemul informatic poate fi considerat dintr-un dublu punct de vedere: ca model extern și ca model intern.

Modelul extern al sistemului informatic se referă la arhitectura

funcţională orientată către utilizator, pe care îl interesează informaţiile şi procesele lor de prelucrare.

Modelul intern al sistemului informatic se referă la structura fizică orientată către echipamente, unde interesează datele, procedurile de prelucrare a datelor şi informaţiilor, echipamentele şi sistemele de operare.

Primul pas în realizarea sistemului informatic constă în definirea modelului extern care va determina modelul intern. Din punct de vedere funcţional, sistemul informatic poate fi conceput ca un ansamblu de subsisteme/componente funcţionale, care sintetizează cerinţele informaţionale ale fiecărei funcţii specifice agentului economic precum şi fiecărui nivel din structura managementului (strategic, tactic şi operativ). Putem constata că specificarea funcţiilor subsistemelor şi aplicaţiilor din structură constituie un moment de decizie majoră în realizarea sistemului informatic [94, 11].

Criteriile de fundamentare ale modelului extern sînt:
- utilitatea, în raport cu obiectivele agentului economic;
- fezabilitatea echipamentelor şi procedurilor;
- eficienţa, care cere recuperarea cheltuielilor de realizare, exploatare şi întreţinere.

Criteriile cele mai semnificative pentru definirea arhitecturii sistemului informatic sînt modularitatea şi integrarea.

Modularitatea se referă la identificarea entităţilor funcţionale temporal independente, între care există interfeţe minime din punctul de vedere al transferului datelor şi informaţiilor.

Integrarea componentelor sistemului informatic se realizează prin date şi prin procedurile de prelucrare.

Integrarea prin date constă în aceea că diferitele subsisteme şi aplicaţii vor utiliza un fond comun de date (baza de date), fără a se exclude şi existenţa unor colecţii de date proprii, specifice. Acest tip de integrare prezintă avantajul că datele despre fenomenele şi procesele petrecute într-un subsistem şi introduse în baza de date pot fi cunoscute şi folosite în celelalte subsisteme şi aplicaţii care au acces la aceeaşi bază de date.

Practica a demonstrat că, totuşi, principiul bazei de date nu trebuie absolutizat, deşi se recomandă ca baza de date să conţină toate datele din sistem, înregistrate o singură dată. Uneori frecvenţa de utilizare a datelor, sursa şi destinaţia informaţiilor, gradul de securitate ce trebuie asigurat şi gradul de permanenţă sînt factori care trebuie luaţi în seamă

la definirea fondului comun de date.

Integrarea prin proceduri constă în cuplarea mai multor aplicaţii, care corespund din punct de vedere funcţional unor activităţi din sectoare diferite, într-o singură aplicaţie, în scopul reducerii numărului de interfeţe şi a numărului de intrări. De exemplu, aplicaţia de gestiune a stocurilor de materiale interesează şi compartimentul de aprovizionare, şi cel de producţie, şi cel de desfacere, financiar-contabil sau de planificare.

Conform acestor modalităţi de integrare, fiecare subsistem poate utiliza colecţii de date (fişiere) proprii, după cum pot exista colecţii de date comune mai multor subsisteme, organizate în baza de date a întregului sistem informatic. De asemenea, subsistemele pot avea proceduri proprii sau cuplate în module folosite în comun de către mai multe subsisteme.

Sistemul informatic, văzut ca sistem suport pentru decizii poate conţine şi modele analitice şi de decizie care, şi ele, pot fi utilizate în comun de către mai multe subsisteme (modele de planificare, de simulare, analiză, statistică etc.).

În condiţiile de mai sus, arhitectura funcţională a unui sistem informatic integrat se prezintă ca în fig. nr. 1.1.

Pe baza unei asemenea concepţii de arhitectură funcţională se stabileşte modelul intern (structura fizică a sistemului informatic), în speţă echipamentele necesare, structurile de date, suporţii pentru stocarea datelor şi informaţiilor, modurile de exploatare, procedurile necesare, personalul de exploatare şi întreţinere ş.a.m.d.

Conceptul de sistem informatic integrat a deschis o etapă calitativ superioară în evoluţia sistemelor informatice. Acest concept aduce pe prim plan rolul informaţiei ca substanţă de ordin superior şi ca resursă a cărei gestionare eficientă constituie un obiectiv distinct în perfecţionarea sistemelor informatice.

Conceptul de integrare are corespondent în tehnologia organizării datelor în baze şi bănci de date, baze de cunoştinţe, care asigură avantaje cum sînt:

- timp minim de răspuns;
- redundanţă minimă în colecţia de date şi în proceduri;
- independenţa procedurilor faţă de organizarea datelor şi cunoştinţelor.

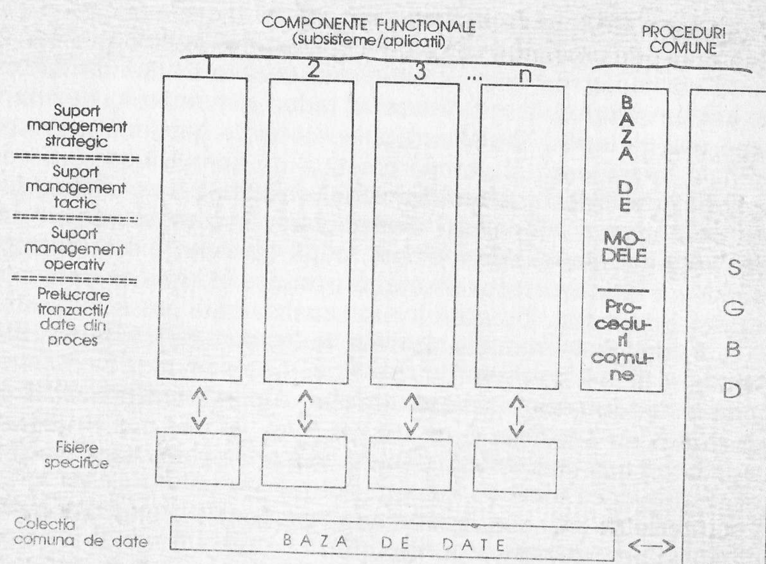


Figura 1.1. Arhitectura funcțională a sistemului informatic integrat

1.2. Mutații actuale în concepția și tehnologia sistemelor informatice

Principalele mutații în planul concepției și tehnologiei sistemelor informatice sînt: aplicarea principiului integrării, aplicarea principiului modularizării, aplicarea principiului ierarhizării, aplicarea informaticii distribuite și introducerea sistemelor bazate pe cunoștințe (knowledge based systems). Să tratăm pe rînd aceste mutații în continuare.

Aplicarea principiului integrării asigură atît legăturile interne dintre componente (integrarea orizontală) cît și conexiunile cu alte componente situate pe nivelurile superioare de management (integrarea verticală). Astfel, sistemul informatic al unui agent economic va asigura un flux de informații conform regulilor stabilite de management, atît în interior cît și în exterior în sensurile prestabilite.

Între toate componentele sistemelor informatice integrate trebuie asigurată compatibilitatea metodologică, prin utilizarea acelorași sisteme de codificare (nomenclatoare unice de coduri) precum și a unui sistem de indicatori economici și tehnici unitari.

Aplicarea principiului modularizării vizează îmbunătățirea procesului de realizare a sistemului informatic dar și valorificarea integrală a resurselor, prin faptul că fiecare modul funcționează ca o entitate cu o anumită arie de cuprindere și cu elemente cu diverse viteze de lucru și timpi de răspuns. Cel mai lent acționează factorul uman în cadrul dialogului precizat în documentația de exploatare.

Aplicarea principiului ierarhizării se datorează caracteristicilor proceselor decizionale, care se subordonează în scară în funcție de nivelurile managementului, în sensul că deciziile organelor superioare sînt obligatorii pentru cele inferioare iar deciziile și normativele organelor inferioare trebuie să se încadreze în limitele stabilite de organele de decizie superioare. Rezultă astfel, în mod obligatoriu, din punct de vedere funcțional și fizic, o schemă conceptuală de sistem informatic ierarhizat, cu mai multe niveluri.

Aplicarea informaticii distribuite favorizează prelucrarea datelor și corectarea erorilor la locul de generare a acestora, lucru posibil prin răspîndirea rețelelor de calculatoare prevăzute cu terminale de toate tipurile și/sau microcalculatoare în rețea și concentratoare de date.

Introducerea sistemelor bazate pe cunoaștere constituie cea mai modernă tehnologie informatică, prin care produsele informatice sînt dotate cu inteligență artificială. La această tehnologie de vîrf este de cîțiva ani sensibilă și contabilitatea, în contextul aplicațiilor sistemelor expert în domeniul economic.

1.3. Probleme ale contabilității manageriale și cerințele sistemului informatic

Elementele sistemului informatic al contabilității constituie nucleul sistemelor informaționale moderne, afirmă Ronald I. Thacker în lucrarea sa "Introduction to Modern Accounting", autor care examinează contabilitatea din punctul de vedere al utilizatorilor informațiilor sale. Considerăm că nu este lipsită de importanță, pentru stadiul actual al economiei românești, prezentarea succintă a rolului contabilității manageriale în viziunea acestui autor.

Contabilitatea are un rol indispensabil în activitatea de management. Pentru a cunoaște mai bine rolul contabilității în sfera sa de acțiune, este esențială în primul rînd o diferențiere între activitatea de procurare a informațiilor necesare procesului decizional din interiorul întreprinderii și

activitatea de procurare a informaţiilor din exteriorul ei.

Acest criteriu stă la baza constituirii a două ramuri ştiinţifice cu caracter contabil: contabilitatea managerială (management accounting) şi contabilitatea financiară (financial accounting).

Contabilitatea managerială are ca obiectiv principal procurarea informaţiilor contabile din interiorul întreprinderii în scopul utilizării lor numai de către consiliul de administraţie şi celelalte niveluri de management ale aceluiaşi agent economic.

Contabilitatea financiară are ca obiectiv procurarea informaţiilor provenind din exteriorul întreprinderii, în vederea utilizării lor de către management şi de către terţi.

Rezultă că, activităţile şi lucrările specifice contabilităţii manageriale interesează numai întreprinderea şi nu se desfăşoară pe baza unor norme unitare şi reguli general acceptate, în timp ce activităţile şi lucrările contabilităţii financiare sînt normate şi îndrumate în mod unitar pe economie, deoarece informaţiile sale sînt considerate de un fel mai special, ele interesînd deopotrivă alţi agenţi economici, guvernul sau persoanele particulare (acţionarii), pe linia situaţiilor financiare de sinteză: bilanţul, situaţia rezultatelor financiare şi contul de "Profit şi pierderi", care reflectă schimbările intervenite în poziţia financiară. Pentru această ramură a contabilităţii există principii general acceptate, care se referă la numeroase concepte şi relaţii ale contabilităţii, dezvoltate în timp şi de mare folos în normarea contabilităţii.

Utilitatea contabilităţii se poate privi din trei unghiuri de vedere:

a) Cea mai evidentă utilitate este pentru managementul întreprinderii, căruia îi furnizează zilnic (chiar şi în timp real) informaţii din cinci grupe de activităţi majore: asigurarea cu mijloace materiale şi băneşti, investiţiile, activitatea de producţie, activitatea de desfacere şi activitatea de distribuire a rezultatelor financiare. Se înţelege că un sistem de contabilitate modernă procură informaţii din toate aceste activităţi, în scopul luării deciziilor. De aici vitalitatea contabilităţii pentru luarea deciziilor pentru oricare tip de agent economic;

b) La fel de importantă este contabilitatea şi pentru utilizatorii externi - acţionari, creditori, guvernul, angajaţii şi clienţii întreprinderii. Aceste categorii de utilizatori sînt interesaţi asupra situaţiei economico-financiare a întreprinderii pentru a se putea decide în multiplele acţiuni de utilizare a mijloacelor financiare de care dispun sau la care au dreptul şi chiar pentru a observa dacă întreprinderea nu desfăşoară practici care să contravină intereselor lor;

c) Pentru societate, contabilitatea prezintă o mare utilitate în privinţa asigurării setului de informaţii pentru controlul întreprinderii, a perspectivei de dezvoltare, asigurarea unui standard de viaţă, mobilizarea de rezerve la nivelul naţiunii şi alte decizii de interes public.

În viziunea aceluiaşi autor, componentele sistemului de prelucrare pe cale informatică a datelor contabilităţii sînt:

- identificarea operaţiilor (tranzacţiilor) economico-financiare şi pregătirea documentelor sursă;
- conversia datelor pe suporţi şi introducerea în sistem;
- prelucrarea internă, cu repartizarea datelor automat pe conturi, calculul valorilor şi memorarea informaţiei contabile obţinute;
- editarea rapoartelor pentru management şi selecţia informaţiilor necesare utilizării viitoare;
- transmiterea rapoartelor şi situaţiilor financiare către utilizatorii din interiorul şi exteriorul întreprinderii.

Potrivit cu aceste cinci componente, un sistem integrat de contabilitate întîi trebuie să identifice automat orice operaţie economico-financiară, să o analizeze dintr-un punct de vedere dual, în termeni de Debit şi Credit, repartizînd sume asupra conturilor care intră în corespondenţă, şi să prelucreze datele selectate cu înaltă precizie pentru pregătirea rapoartelor şi situaţiilor financiare necesare tuturor nivelurilor de decizie. Orice operaţie are la bază un document sursă, care conţine datele detaliate. Documentele sursă trebuie proiectate în aşa fel încît să ofere o bună relevanţă a operaţiei economico-financiare. Pe baza lor se pregătesc intrările în sistemul informatic al contabilităţii. Orice document sursă trebuie să conţină cel puţin contul sau conturile de debitat, suma sau sumele ce se trec în debit, contul sau conturile de creditat şi suma sau sumele ce se trec în credit. Atunci cînd managementul solicită şi alte informaţii în afara celor cu caracter strict contabil, documentele sursă vor conţine datele adecvate.

Conversia pe suporţi şi introducerea datelor în sistemul informatic integrat al contabilităţii presupune folosirea unor proceduri speciale precum şi echipamente de calcul corespunzătoare, dotate cu software.

Prelucrarea internă se referă la prelucrarea pe baza procedurilor automate a datelor contabile, elaborate pe baza tehnicii contabile curente şi principiilor general recunoscute. Toate operaţiile de calcul şi editare se desfăşoară automat, obţinîndu-se în final ieşirile concretizate în diverse rapoarte pentru management şi situaţii financiare, precum şi alte informaţii care se memorează în vederea unor prelucrări sau utilizări

viitoare.

Tema centrală pentru orice sistem modern de contabilitate nu este, oricît apare de vizibil, simpla prelucrare a datelor, ci este procesul complex de selecție a informațiilor prelucrate după principiile contabilității, atașarea valorilor la fiecare tranzacție economică și pregătirea rapoartelor semnificative pentru luarea deciziilor.

În managementul modern, un număr mare de decizii se referă la previziunea rezultatelor financiare iar informațiile contabilității ajută cel mai bine în acest sens. Managementul are sarcina și responsabilitatea să găsească căile de obținere a celor mai bune rezultate, să îmbunătățească veniturile de la o perioadă la alta. O cale de acțiune permanentă este analiza și compararea elementelor care formează venitul net, ca sumă absolută și în procente pe ani.

Tot echipa de management trebuie să analizeze schimbările intervenite în volumul și ponderea cheltuielilor, determinînd efectul lor asupra veniturilor, să compare venitul net pe fiecare perioadă cu cea precedentă din aceeași unitate și din unități similare.

Venitul net se determină în procesul de calcul contabil prin identificarea, măsurarea și compararea veniturilor brute cu cheltuielile dintr-o anumită perioadă de gestiune:

VENIT BRUT - CHELTUIELI = +/- VENIT NET

Această relație este considerată fundamentală alături de A = P.

Metodele de determinare a costurilor precum și gruparea veniturilor și evaluarea lor sînt importante pentru obținerea venitului net.

Contabilitatea elaborează bugetul de venituri și cheltuieli pentru o perioadă dată. Pentru aceasta se folosește de previziuni și face toate estimările necesare.

Obiectivele contabilității sînt mai clare dacă sînt puse față în față cu obiectivele managementului, pe aspectele legate de mijloace și resurse, pe elementele de activ și pasiv.

Mijloacele și resursele sînt absolut vitale pentru desfășurarea activității și trebuie ținut seama la proiectarea sistemului informatic integrat, de cea mai importantă categorie de resurse - resursele umane. În legătură cu acestea, contabilitatea trebuie să raporteze sistematic schimbările intervenite, costul procesului de selecție, pregătire, menținere și înlocuire.

Tehnica contabilă se desfășoară în conformitate cu principiile general acceptate ale contabilității, care au ca punct de plecare, în mod obligatoriu, înregistrarea mijloacelor și resurselor sau egalitatea Activ = Pasiv și pe baza ei înregistrarea schimbărilor intervenite în

elementele de activ și de pasiv. Cum pot fi măsurate elementele de activ și de pasiv, ce informații trebuie dezvăluite cu privire la acestea, ce situații financiare se pregătesc managementului și ce informații pentru terți, iată problemele care au fost soluționate în timp și care s-au cimentat în mod organic într-o teorie pe baza căreia se desfășoară aplicațiile contabilității. "Încercările de a fundamenta deciziile pe baza altor informații decît cele oferite de contabilitate au eșuat în mod lamentabil" - spune autorul la care ne-am referit.

1.4.Privire sistemică asupra contabilității

O punere în lumina concepției sistemice a devenit astăzi obligatorie pentru oricine studiază perfecționarea sau modernizarea contabilității, indiferent de metoda critică adoptată, tematică sau formală. Ideile precursoare de modernizare a contabilității fac referiri la contabilitatea ca sistem. În condițiile în care cibernetica va condiționa întreaga activitate a mileniului viitor se înțelege de ce facem și noi o asemenea încercare. Intenția de a trata din punct de vedere sistemic contabilitatea, trebuie înțeleasă deci, în contextul deschiderilor oferite de aplicarea principiilor ciberneticii în domeniul contabilității ca știință și practică.

Concepția sistemică ne cere să identificăm obiectivele contabilității ca sistem, subsistemele sau elementele componente, intrările și ieșirile, conexiunile directe și inverse dintre componente, precum și conexiunile cu mediul, structura și legile care determină comportarea dinamică și funcționarea fiecărei componente. Analiza sistemică reprezintă totodată și metoda de bază pentru elaborarea scenariilor de dezvoltare a contabilității pentru aplicarea celor mai importante tehnici și identificarea contribuției activității financiar-contabile la realizarea în condiții de optim a funcționării agenților economici. De fapt, privirea sistemică a contabilității a devenit un act de identitate intelectuală, prin care introducerea tehnologiilor de vîrf, ca semn al epocii, se va dovedi mai activă în acest domeniu al științei și, se înțelege, inclusiv pentru dezvoltarea sistemelor informatice integrate.

O concepție originală sub raport științific este aceea a cercetătorului Rath S. Sudhansu, de la California State University din Los Angeles. Menționăm în continuare principalele sale idei.

Contabilitatea este un sistem socio-tehnico-economic, foarte mult influențat de mediul economic și social. Obiectivul de bază al contabilității

ca sistem, este obţinerea situaţiilor financiare (Bilanţul şi celelalte), care reprezintă adevărata stare economico-financiară a întreprinderii. Contabilitatea este un sistem cu retroacţiune (conexiune inversă), ale cărui componente sînt în număr de şapte: intrările, ieşirile, procesorul, filtrele, standardele, comparatorul şi unitatea de control (vezi fig. 1.6.)

INTRĂRILE în sistemul contabilitate sînt de două feluri de bază: date şi instrucţiuni. Datele se referă la operaţiile economico-financiare şi planul agentului economic. Instrucţiunile constau în principii, tehnici şi instrumente. Toate acestea sînt formulate de agenţii sau instituţii specializate pentru exercitarea profesiei de economist contabil în rezolvarea problemalor de contabilitate într-o manieră unitară şi uniformă. Instrucţiunile fac obiectul celor mai frecvente perfecţionări şi se regăsesc întocmai în sistemele informatice contabile.

IEŞIRILE se pregătesc de către procesor pe baza instrucţiunilor, obţinîndu-se situaţiile financiare.

PROCESORUL transformă intrările în ieşiri, folosind echipamente adecvate şi personal de specialitate. El are legătură imediată cu instrucţiunile.

FILTRELE constituie un concept util deoarece se concretizează într-o persoană specializată sau alt factor din mediul exterior, avînd sarcina să valideze elementele sistemului. Pentru sistemul contabilitate, filtrele controlează alimentarea procesorului numai cu intrările prestabilite şi respectiv reîntoarcerea informaţiei (retroacţiunea) de la comparator, înaintea sosirii sale la unitatea de control.

STANDARDELE conţin norme clare, unitare privind conţinutul situaţiilor financiare necesare pentru a fi comparate cu situaţiile elaborate de procesor la ieşire. Ele se referă la ultima versiune unanim acceptată şi aplicată de către toţi agenţii economici. Şi acestea sînt supuse deseori perfecţionării în conformitate cu cerinţele formulate de către mediul economico-social (vezi adaptarea legislaţiei contabilităţii pentru economia de piaţă).

COMPARATORUL este esenţial, alături de standarde, în oricare sistem, în scopul măsurării eficienţei. În cazul sistemului contabilitate, mediul economico-social serveşte în calitate de comparator, pentru că acest sistem este în serviciul societăţii iar contabilitatea este foarte mult influenţată de mediul social-economic (guvern, acţionari, agenţi economici). Acest mediu este cel mai bun judecător al eficienţei şi eficacităţii contabilităţii. El compară standardele cu ieşirile. Criteriile de comparare sînt stabilite de către mediul însuşi iar natura criteriilor se poate schimba

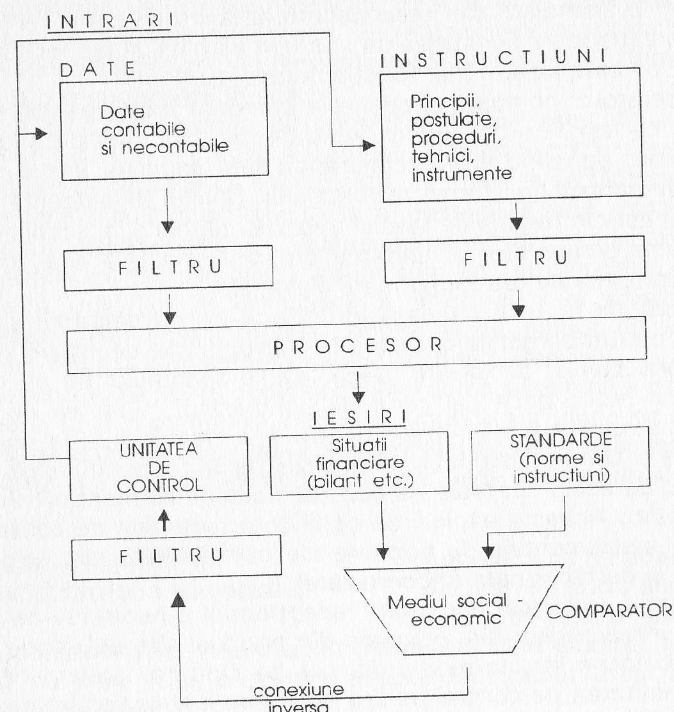


Figura 1.6. Schema sistemului «contabilitate»

în timp inclusiv în sensul simplificării.

UNITATEA DE CONTROL se constituie din agenţiile naţionale sau instituţiile care elaborează norme şi metodologii unitare pentru organizarea şi conducerea contabilităţii. Funcţia sa este de a menţine nivelul cerut de calitate pentru ieşirile din sistem.

Înţelegerea funcţionării sistemului contabilitate este importantă pentru desprinderea direcţiilor de perfecţionare a contabilităţii. Înainte de a intra în procesor, datele de intrare sînt controlate de filtru, eliminîndu-se nepotrivirile sau datele neoportune contabilităţii. Procesorul, după ce primeşte datele de intrare, le grupează şi le prelucrează în conformitate cu instrucţiunile de intrare, generînd situaţiile financiare prestabilite.

Situaţiile financiare sînt recunoscute la ieşire de către comparator. Această presupunere constituie un postulat de bază al teoriei comunicaţiilor, fără de care sistemul ar funcţiona neeficient.

Comparatorul compară şi cerinţele standard fixate pentru situaţiile financiare cu ieşirile procesorului, luînd decizia dacă aceste situaţii sînt bune sau nu, generînd erorile constatate prin semnale adecvate către unitatea de control sub forma retroacţiunii. Nivelul de exigenţă al comparatorului este în funcţie de stadiul evoluţiei teoriei şi aplicaţiilor contabilităţii. Între comparator şi unitatea de control există acel fitru care elimină informaţia de retroacţiune nerelevantă.

Unitatea de control studiază informaţia de retroacţiune şi trimite comenzile pentru corecţiile necesare în instrucţiunile de intrare şi uneori chiar asupra datelor de intrare. Dacă ieşirile sistemului au un grad de precizie mai redus decît cel cerut de comparator, unitatea de control schimbă instrucţiunile de intrare pentru a aduce ieşirile la nivelul de precizie dorit. Un grad mai înalt de precizie decît cel cerut indică o risipă de resurse, de aceea unitatea de control încearcă să menţină nivelul de precizie optim. Aceasta explică de ce şi cum metodele de contabilitate practică şi instrumentele de normare ale contabilităţii sînt schimbate, modificate şi perfecţionate (modernizate).

Informaţia de retroacţiune este recepţionată de unitatea de control sub forma intervenţiilor din articolele din presă şi alte publicaţii, conferinţe, simpozioane etc. în mod continuu, pe cînd comanda de corecţie primită de unitatea de control pentru ajustarea instrucţiunilor de intrare este periodică şi-i exprimată sub forma unor modificări în procedurile, tehnicile şi instrumentele specifice practicii contabile, precum şi în lucrările de normare a contabilităţii în general.

Eficienţa sistemului depinde de trei factori: eficienţa fiecărei componente, comunicaţia între componente şi capacitatea fiecărei componente de a se adapta la noile condiţii impuse de mediu. Întrucît componentele sistemului sînt toate de natură economică, graniţele fiecărei componente trebuie foarte bine delimitate.

În privinţa colecţiei datelor de intrare, costurile de colectare şi stocare în raport cu utilitatea lor trebuie luate în seamă, deoarece de cele mai multe ori pot fi considerabile.

Instrucţiunile de intrare trebuie să aibă o balanţă optimă între flexibilitate şi uniformitate. Persoanele care aparţin uneia din cele mai importante componente - comparatorul, trebuie instruite să înţeleagă scopul situaţiilor financiar-contabile şi modul lor efectiv de utilizare.

Responsabilitatea pentru instruirea lor trebuie să o aibă specialiştii în contabilitate cu educaţie superioară. Filtrele folosite în sistem trebuie să fie complet nepărtinitoare. Un filtru părtinitor deformează informaţia care trece de la o componentă la alta.

Viteza cu care unitatea de control ia decizii este, de asemenea, determinantă pentru eficienţa sistemului. Eficienţa oricărui sistem este mult afectată de reţeaua de comunicaţii. Ea depinde de factori cum sînt: viteza cu care circulă informaţia de la un element la altul, capacitatea de percepţie a informaţiei de către fiecare componentă, periodicitatea la care componentele constituie cauza informaţiei de retroacţiune, viteza de percepţie a intrărilor şi prelucrare a informaţiei la nivelul fiecărei componente. Toţi aceşti factori condiţionează timpul de răspuns şi implicit contribuţia la maximum de eficienţă al sistemului.

În concluzie, se poate observa că abordarea sistemică a contabilităţii este o temă utilă şi complexă, deoarece implică atît teoria cît şi practica contabilă, precum şi activitatea instituţiilor de control şi îndrumare a contabilităţii la scara întregii economii şi la nivel internaţional.

1.5. Scenarii privind contabilitatea viitorului.

Apreciem că nu poate fi lipsită de interes pentru scopul materialului de faţă, o sinteză referitoare la contabilitatea viitorului, în momentul cînd gîndirea ştiinţifică a atins cel mai înalt nivel, provocînd asupra domeniului nostru de investigaţie salturi de ordin superior.

Mai întîi, în planul teoriei trebuie să observăm instaurarea unui mod de gîndire integral dinamic, în care se aplică în contabilitate principiile teoriei generale a sistemelor, a ciberneticii şi metodele informaticii, în scopul conferirii unor orizonturi mai largi de cuprindere a obiectului de studiu şi consolidării definitive a metodei de cercetare. (vezi Andone, I., "Modernizarea contabilităţii întreprinderilor din ramura metalurgiei fierului", 1985, p. 80 ş.u.)

Am observat că se pot desprinde şi idei novatoare, de restructurare a practicii contabile, care în mod cert în viitor va fi complet informatizată, iar îndrumările şi normele de organizare şi conducere a activităţii de contabilitate vor fi mereu adaptate cerinţelor managementului.

Actualele relaţii poliaxiale instituite la scara întregii societăţi, informaţie-creştere economică, informaţie-educaţie, determină mai buna utilizare în viitor a sistemului informaţional-contabil de către manage-

mentul strategic, tactic și operativ, fapt care va impune desfășurarea în timp real a procesului de informare contabilă și consacrarea definitivă a teoriei și aplicațiilor contabilității dinamice și previzionale.

Contabilitatea viitorului va trebui să aibă mereu pregătite informații de previziune, care să orienteze managementul în deciziile viitoare. Pentru aceasta va face analize ex ante. O structură de viitor a contabilității va trebui să fie mai normativă și mai analitică decît în trecut [Goldsmidt, Y., 4].

Creatori de știință în contabilitate își vor concentra efortul și spre generalizarea formalizării matematice pentru fundamentarea tuturor procedurilor informatice contabile, postfaptice și de previziune, în concordanță cu criteriile de economisire a resurselor sistemelor informaționale, în principal a resurselor umane implicate.

Punctul forte al reflecției contabile rămîne în continuare practica din spațiul activității economico-sociale, efectuarea ei pe baza unor criterii de maximă raționalitate într-un ritm care să satisfacă nevoia socială după criterii multiple. Aceasta presupune dezvoltarea deosebită a contabilității manageriale, al cărei nucleu, evidența analitică a cheltuielilor de producție și calculul costurilor, va constitui componenta centrală a sistemului informațional integrat al agentului economic, în întregime informatizat.

Fondul de date al contabilității va fi gestionat în mod unitar în cadrul băncii de date a sistemului informațional automatizat iar informațiile esențiale pentru dezvoltarea contabilității ca sistem se vor administra după sistemul bazei de cunoștințe conform tehnologiei sistemelor expert.

Cererile compartimentului contabilității și/sau compartimentului financiar se vor satisface după o concepție unitară, cu utilizarea celor mai moderne tehnologii informatice, în principal pentru controlul bunei funcționări a sistemului informatic contabil și pentru obținera situațiilor de sinteză, analiză și previziune economico-financiară.

Informațiile financiar-contabile care vor servi managementului tactic și operativ vor fi primite automat (periodic și/sau la cerere), în momentul solicitării și la punctele de decizie unde sînt necesare, concatenate dacă este cazul cu informații provenite din alte subsisteme.

*
* *

Din practica realizării sistemelor informatice a rezultat că sprijinul economiștilor contabili este indispensabil, pe toată durata concepției și elaborării procedurilor de contabilitate. Informațiile cu caracter contabil au periodicități proprii, asupra cărora soluțiile experților din domeniu trebuie respectate cu rigurozitate. Pe acest plan al adevărului, teoria ne pune în fața unor necesități care direcționează actualele preocupări de a integra în pregătirea viitorilor specialiști, toate cunoștințele necesare de informatică economică alături de cunoștințe largi despre disciplină, cele de ordin tehnic privitoare la stăpînirea matematicilor, statisticii, teoriei micro și macroeconomice și financiare.

CAPITOLUL II

BAZELE TEORETICE ALE INTELIGENŢEI ARTIFICIALE

2.1. Inteligenţa artificială şi prelucrarea automată a datelor.

Integrarea şi inteligenţa sînt caracteristicile de viitor ale sistemelor informatice contabile. Inteligenţa artificială şi cercetarea, dezvoltarea şi introducerea sistemelor de inteligenţă artificială au cunoscut o extindere rapidă în ultimul deceniu.

Inteligenţa artificială, numită uneori şi inteligenţa maşinii sau programare euristică, reprezintă o ştiinţă precum şi o tehnologie de vîrf, care a atras un număr mare de cercetători, determinînd realizări de prestigiu şi multe aplicaţii în cele mai diverse domenii.

O simplă trecere în revistă a acestei discipline ne relevă că inteligenţa artificială este în legătură cu elaborarea de programe, care să rezolve probleme din orice domeniu la nivelul expertului uman şi să comunice rezultate în limbaj natural.

Cercetările în inteligenţa artificială sînt axate pe abordarea cu ajutorul calculatoalor electronice a comportamentului inteligent al sistemelor de orice tip. Se au în vedere două obiective: înţelegerea comportamentului inteligent şi producerea de maşini inteligente, mai utile societăţii decît cele ale prezentului.

Programele calculatoarelor cu care inteligenţa artificială are legătură directă, desfăşoară adevărate **prelucrări simbolice**, care implică incertitudinea, ambiguitatea şi complexitatea. Asemenea prelucrări sînt necesare atunci cînd nu este nevoie sau nu există soluţii algoritmice pentru rezolvarea unei probleme dintr-un domeniu specializat. În jargonul

inteligenţei artificiale **simbolul** este un şir de caractere pentru un concept din lumea reală. Simbolurile pot fi combinate pentru exprimarea relaţiilor dintre ele. Cînd aceste relaţii sînt reprezentate într-un program de inteligenţă artificială, ele se numesc structuri de simboluri. De exemplu:

(DEFECT produsul)
(ÎNCHIRIAT celui care l-a defectat)
(SUMA (DATORIA vinovatului) 1000)

Această structură de simboluri se interpretează astfel:
- produsul este defect;
- produsul a fost închiriat celui care l-a defectat;
- suma stabilită ca datorie a vinovatului este de 1000 lei.

Fig.2.1. Tablou comparativ între inteligenţa artificială şi programele convenţionale

Progamele de inteligenţă artificială	Programele convenţionale p.a.d.
1. Obiectiv: rezolvarea unei probleme prin prelucrarea cunoaşterii;	1. Obiectiv: obţinerea informaţiilor prin prelucrarea datelor;
2. Execută prelucrarea simbolică a cunoaşterii înmagazinată în baze de cunoştinţe, folosind limbaje simbolice;	2. Execută îndeosebi prelucrarea numerică a datelor din fişiere şi baze de date;
3. Căutare euristică (soluţia este obţinută prin inferenţă logică);	3. Algoritmi (soluţia este obţinută prin operaţii explicite, codificate în limbaje procedurale);
4. Structura de control este separată de cunoaştere şi este introdusă în "motorul de inferenţe";	4. Structura de control şi datele sînt integrate în ceea ce se numeşte procedură;
5. Sînt uşor de modificat, actualizat şi extins prin dialog în limbaj natural;	5. Sînt dificil de modificat cu instrumente de programare;
6. Tolerează şi răspunsuri aproximative;	6. Cer numai răspunsuri exacte;
7. Produc şi soluţii uzual acceptabile.	7. Se cer de la ele numai soluţii perfecte.

Inteligența artificială este în legătură cu rezolvarea problemelor de luare a deciziilor, care preocupă oamenii în mod continuu în mediul economico-social. Tehnologia specifică inteligenței artificiale diferă de cea a prelucrării automate a datelor, care-i, în principal de natură numerică. Prin contrast, programele de inteligență artificială operează asupra conceptelor, faptelor și situațiilor, oferind răspunsuri asemănător modului de soluționare umană a problemelor. Și ca obiectiv diferă cele două tehnologii (fig. 2.1.). Tehnologia inteligenței artificiale urmărește rezolvarea unor probleme prin prelucrarea cunoașterii, în timp ce tehnologia clasică, convențională, urmărește obținerea de informații în urma prelucrării datelor.

Caracteristica de bază a inteligenței artificiale este **căutarea euristică.** În problemele complexe, numărul soluțiilor posibile poate fi enorm, de aceea rezolvarea problemelor pe baza tehnologiei inteligenței artificiale este în mod obișnuit orientată după reguli empirice, referitoare la euristici (intuiție, experiență, generalizare), folosindu-se intens cunoștințele acumulate în domeniu. Inteligența programelor este puternic dependentă de cantitatea cunoștințelor disponibile în baza de cunoștințe și structurile de control ale cunoașterii introduse în motorul de inferențe.

Cunoștințele trebuie să fie disponibile pentru utilizare atunci cînd sînt necesare, în timpul căutării. Pe această cale, schimbările în cunoaștere impun schimbări numai în baza de cunoștințe. Prin contrast, datele problemei și structurile de control în programele algoritmice convenționale sînt integrate. Ca urmare aceste programe sînt mult mai dificil de modificat, tocmai pentru că schimbările într-o parte a programului implică schimbări și în celelalte, chiar dacă sînt concepute modular.

2.2. Privire istorică asupra inteligenței artificiale

Inteligența artificială își are originea în dezvoltările noi pe baza științelor de graniță, informatica, teoria sistemelor, cibernetica, logica matematică, lingvistica matematică, teoria cunoașterii, psihologia, fiziologia creierului, teoria deciziei, teoria informației, teoria automatelor ș.a., toate în legătură cu fundamentarea relațiilor dintre om și tehnică, dintre om și mașină.

Perioada 1936 - 1956 reprezintă începutul unor asemenea studii. Savantul român Ștefan Odobleja, creatorul ciberneticii, între anii 1937 - 1939 a prevăzut apariția inteligenței artificiale și rolul ei în creșterea productivității muncii. Sînt deosebit de interesante lucrările savanților A.Newell, I.C.Show, H.A.Simon, A.Turing, C.E.Shannon, J.McCarty, M.Minsky, Gr.Moisil, N.Wiener, L. von Bertallanfy, N.J.Nilsson, T.Winograd, E.Hunt ș.a..

Conjuncția dintre științele economice și inteligența artificială a constituit un puternic factor de declanșare a unor noi cercetări, pentru rezolvarea problemelor de decizie și organizare complexe, care au devenit domenii de proiectare inteligentă, mai mult decît al aplicării tehnicilor clasice de optimizare. Astfel s-a transformat paradigma "Homo-Economicus" în "Homo-Cogitans", care va domina introducerea raționamentului economic într-un mediu care să-i permită să inventeze și să dezvolte noi forme de inteligență, pentru satisfacerea după criterii multiple a preferințelor umane.

În anul 1956, un mic grup de oameni de știință au hotărît organizarea unei conferințe la Dartmouth, unde să se facă publice unele rezultate în domeniul inteligenței artificiale, prilej cu care s-a lansat și termenul de inteligență artificială[1].

Predicția elaborată atunci se referea la faptul că, în următorii 25 de ani, inteligența artificială se va implica în toate activitățile umane în care calculatoarele vor lucra intens, obținîndu-se produse de inteligență artificială sau sisteme de inteligență artificială[2]. Anul 1956 mai este semnificativ și din punctul de vedere al prezentării programului LT (Logical Theorist) al savanților A.Newll, J.C.Show și H.A.Simon, program care demonstra teoreme peste nivelul mediu de inteligență. Programul s-a remarcat prin nivelul prelucrării informațiilor prezentate și memorate în formă simbolică, fapt care a demonstrat că gîndirea simbolică poate fi exprimată în termenii problemelor de rezolvat iar pe baza cunoștințelor din domeniul aplicativ se poate acționa cu programe inteligente. Din acest moment realizările inteligenței artificiale se succed cu repeziciune.

În anul 1972, M.Minsky introduce conceptul de "frame", un șablon foarte general (cadru), pentru descrierea proprietăților obiectelor, fenomenelor, situațiilor etc. Ulterior s-au dezvoltat rețelele semantice,

---

[1]Termenul de inteligență artificială derivă din latinescul „intellegere" = a alege dintre, deci a înțelege, a percepe, a ști să perceapă

[2]Produsul sau sistemul care înțelege, achiziționează, prelucrează și transmite cunoștințe este un produs de inteligență artificială: roboți, sisteme expert etc.

sistemele de reguli de producție, limbajele de programare a inteligenței artificiale ș.a.m.d..

În anul 1981, în Canada, la Vancouver, în cadrul unei conferințe internaționale s-a analizat o listă de activități, în care inteligența artificială a făcut progrese uluitoare: traducere automată, integrare simbolică, jocuri diverse (șah, dame etc.), recunoașterea formelor, demonstrarea automată a teoremelor, înțelegerea vorbirii ș.a.. S-a remarcat că, în deceniul 1970 - 1980, o serie de produse ale inteligenței artificiale au devenit deja comercializabile, fiind înființate companii specializate în astfel de produse.

În anii următori, efectele economice ale unor astfel de produse s-au dovedit favorabile. În special roboții și sistemele expert dedicate, precum și mașinile LISP. Aceasta a însemnat ieșirea din așa numita perioadă a "copilăriei" inteligenței artificiale, în care s-au cristalizat conceptele, metodele și tehnologia ei specifică.

Japonia va scoate în curînd primele reprezentante ale calculatorelor din generația a cincea, care vor utiliza conceptele programării logice (PROLOG).

În prezent se pot procura produse de inteligență artificială, cu aplicații eficiente în toate domeniile și se alocă fonduri substanțiale pentru dezvoltarea unor noi aplicații integrate în sisteme inteligente.

În țara noastră, în octombrie 1981, s-au prezentat principalele realizări românești în inteligența artificială și robotică, în cadrul primului Simpozion național de inteligență artificială de la București, sub patronajul Academiei Române. S-au înmulțit colectivele de cercetare și s-au înființat catedre de specialitate și, mai recent, un institut de specialitate, la Craiova, după modelul a numeroase țări. La nivel internațional există numeroase organisme specializate: American Association for Artificial Intelligence, European Coordinating Committee for Artificial Intelligence, International Joint Conference on Artificial Intelligence, International Federation for Information Processing.

2.3. Concepte de bază ale inteligenței artificiale

Inteligența artificială, ca subdomeniu al informaticii și-a consacrat propiile concepte iar altora le-a conturat mai bine înțelesul, adecvîndu-l scopului acestei discipline științifice.

În cele ce urmează ne vom referi la un număr relativ restrîns de concepte ale inteligenței artificiale, tocmai din cauza spațiului restrîns afectat și scopului materiei noastre. Vom trece succint în revistă conceptele: inteligență artificială, cunoaștere, metacunoaștere, sistem cognitiv, euristică, raționament, inferență, model, problemă, sistem rezolutiv, strategie de control și căutare.

2.3.1. Inteligență artificială.

În literatura de specialitate există o diversitate de definiții pentru conceptul de inteligență artificială. Academicianul Mihai Drăgănescu arată că inteligența artificială este o derivată a inteligenței naturale, care capătă forme concret sociale, în sensul că dezvoltă inteligența socială în mod direct și indirect, ca urmare a intereselor în activitatea economico-socială. Autorul menționat prezintă și relațiile specifice între inteligența naturală, inteligența artificială și inteligența socială [Drăgănescu, M., „Informatica și societatea", 74]. Are în vedere cele trei componente ca pe un tot unitar. Astfel, inteligența socială, prin inteligența artificială, devine și forță de producție directă (de exemplu prin inteligența roboților industriali), iar introducerea informaticii, teleinformaticii și telecomunicațiilor conferă consistență inteligenței sociale. La rîndul ei, inteligența artificială este o prelungire a inteligenței naturale (fig. 2.2).

Același autor afirmă că sistemul inteligenței sociale se constituie din

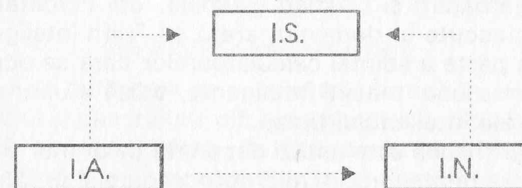


Figura 2.2. Relațiile între inteligența naturală (I.N.), inteligența artificială (I.A.) și inteligența socială (I.S.)

structuri umane, economico - sociale și structuri informatice cu prelucrare inteligentă a informației, începînd de la locul de muncă pînă la nivelurile superioare ale ierarhiei sociale, în condițiile utilizării crescînde a teleinformaticii și telecomunicațiilor, sistemelor expert, roboticii și automatizării.

Dr. ing. Ioan Georgescu, autorul uneia dintre primele cărţi de inteligenţă artificială din ţară, menţionează că, problema definirii inteligenţei artificiale se rezolvă uşor prin preluarea definiţiei inteligenţei umane, ţinînd seama de lipsa de coeziune a unor termeni. El arată că "inteligenţa naturală este facultatea de a înţelege, de a căpăta cunoştinţe, puterea de a face faţă oricărei situaţii, mai cu seamă a celor neobişnuite şi neprevăzute, capacitatea de a reacţiona cu o viteză suficientă pentru a putea corecta atît procesele exterioare cît şi propriul comportament, uşurinţa de a înţelege sau de a descoperi corelaţia dintre obiectele concrete şi cele abstracte, de a opera cu abstracţii, în vederea dirijării acţiunilor către scopul dorit."

Autorii americani A.Barr şi E.Feigenbaum arată că inteligenţa artificială este cel mai interesant şi controversat subdomeniu al ştiinţei calculatoarelor şi studiază posibilităţile de dotare a sistemelor electronice de calcul cu componente (programe şi echipamente) care să presteze activităţi în manieră inteligentă, avînd proprietăţi pe care în mod obişnuit le asociem înţelegerii comportamentului uman cum sînt înţelegerea limbajului natural, învăţarea şi raţionamentul în rezolvarea problemelor.

Sugestivă este şi precizarea lui P.Winston, în sensul că inteligenţa artificială este studiul ideilor care permit calculatoarelor să efectueze acele lucruri care fac pe oameni să pară inteligenţi. Tot acest autor de marcă arată că, obiectivele principale ale inteligenţei artificiale sînt de a face mai utile calculatoarele şi de a înţelege principiile care fac posibilă inteligenţa.

Luca Dan Şerbănaţi şi Cristian Giumale, doi cercetători români cu preocupări recunoscute în domeniu, arată că "prin inteligenţa artificială se înţelege acea parte a ştiinţei calculatoarelor care se ocupă de proiectarea şi construirea unor maşini inteligente, adică a unor maşini care să realizeze funcţii ale intelectului uman."

Inteligenţa artificială este astăzi destinată celor mai diverse tipuri de produse, în care se pot încorpora microprocesoare şi programe adecvate, nu numai domeniului calculatoarelor. Ea are în vedere în mod deosebit "dotarea" acestora cu un comportament inteligent, asemenea omului.

2.3.2. Cunoaştere, metacunoaştere şi sistem cognitiv.

Din punct de vedere ştiinţific este stabilită legătura dintre inteligenţă şi cunoaştere sub aspectul relevării trăsăturilor fundamentale şi a scheletului raţional al cunoştinţelor, pe care ni le

formulăm, într-un cadru adecvat, despre un obiect, fenomen, un eveniment, un proces sau o situaţie, prin efecturea de raţionamente, discernerea generalului, a importanţei şi conturarea ordinii conceptuale prin abstractizarea experienţei şi a particularului.

Cunoaşterea ca noţiune poate fi mai bine înţeleasă dacă este privită din două unghiuri diferite:

- ca un act intelectual, denumit şi observaţie, prin care se stabileşte un transfer de informaţie de la un obiect observat către un subiect cunoscător (fig. 2.3.);
- ca informaţie latentă, descriptivă şi definitorie, depozitată în structurile de memorare ale subiectului cunoscător.

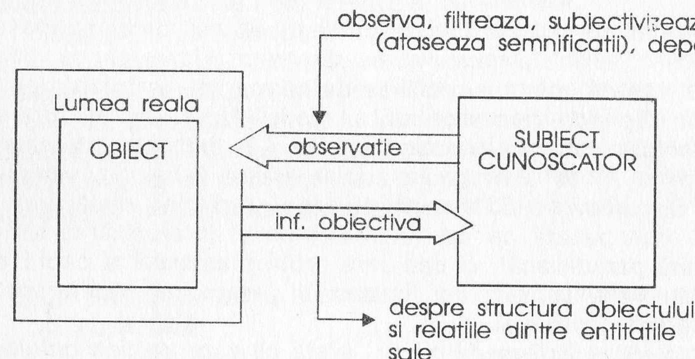


Figura 2.3. Schema cunoaşterii

Cunoaşterea se defineşte ca o reflectare activă în conştiinţă a lumii reale, a esenţialului şi generalului din fenomene, a legăturilor obiective ale realităţii, bazată pe capacitatea de descompunere şi analiză a proprietăţilor, de sinteză a unor obiecte abstracte, inexistente, pornind de la proprietăţi specificate, deci de lărgire a orizontului obiectelor posibile. Ea are două **componente**: una reflexivă (bazată pe reflectarea exterioară, ulterioară a faptului ştiinţific) şi una generativă, constructivă, bazată pe crearea de obiecte abstracte şi pe constituirea de momente iniţiale pentru procesele de construire a unor noi fapte ştiinţifice şi programe de acţiune. Cunoaşterea are o serie de **caracteristici**.

Cunoaşterea este o **categorie filosofică** fundamentală, are o **natură psihologică** deoarece este întotdeauna raportată la subiectul cunoscător;

este **empirică** - în sensul că informația despre realitatea înconjurătoare este sesizată nemijlocit prin organe senzoriale sau prin intermediul aparatelor și instrumentelor specifice: este **teoretică** - dacă se bazazează pe raționamente și judecăți subliniind esența legăturilor interne, cauzalitatea și legitățile care dezvoltă structuri și procese. Cunoașterea teoretică se dezvoltă din cunoașterea empirică, prin analiză și sinteză, inducție și deducție, generalizare și particularizare. Cunoașterea are un **caracter individual**, se justifică din punct de vedere cauzal iar prin teoria științifică obține valoare de adevăr și se obiectivizează.

Cunoașterea despre lume **suportă un continuu proces de rafinare**, ceea ce înseamnă că de fapt cunoașterea este întotdeauna **parțială** sau **incompletă**.

În afară de epistemologie (teoria cunoașterii), de studiul cunoașterii se ocupă numeroase discipline științifice, din propriul lor unghi de vedere. Filozofia însă, are ca obiectiv stabilirea de norme sau standarde pentru reprezentarea faptelor relevante și a modalităților de efectuare a reprezentărilor prin judecăți și raționamente. Un exemplu interesant îl constituie opinia filosofului român Lucian Blaga („Trilogia valorilor", Ed.Minerva, București, 1987), potrivit căreia se disting două tipuri de cunoaștere - cea bazată pe datele observate și interpretările simple, directe (numită paradisiacă), și cea care problematizează și caută descoperirea semnificațiilor, relațiilor, misterelor care se ascund în spatele obiectului (numită luciferică).

Umberto Eco, marele semiotician, arată că este sarcina omului să descopere semnificațiile care se ascund în spatele obiectelor și fenomenelor și de a reconstrui cu ajutorul lor o țesătură a realității numită cunoaștere, deoarece un fenomen sau obiect nu se reprezintă numai pe sine însuși, ci în funcție de relațiile lui cu celelalte obiecte și fenomene din lumea reală, funcție de care reprezintă și un semn[Eco,17].

Pentru scopul pe care îl urmărim în această lucrare, este mai relevantă transpunerea conceptului de cunoaștere de la subiecții cunoscători de tip uman, la subiecții cunoscători de tip program pentru calculator, deci de la psihologia gîndirii la inteligența artificială. Pe acest plan trebuie subliniat că informația latentă, obiectivă, cu caracter de generalitate, în care intră definițiile, legile, regulile, conceptele, clasificările, formează ceea ce denumim cunoaștere. Aceasta este achiziționată ca urmare a unor procese de învățare, observare a realității, abstractizare și obiectivizare prin rafinare permanentă. Pentru a fi utilizată de către un program sau produs-program, cunoașterea este memorată într-o bază de

cunoștințe sub forma unor **piese de cunoaștere**, care descriu faptele, fenomenele, procesele și evenimentele din acea parte a lumii reale care formează domeniul de competență al **programului inteligent**.

Prin intermediul pieselor de cunoaștere se pot reprezenta orice fel de noțiuni relevante pentru un domeniu, este vorba de concepte și instanțe (particularizări ale conceptelor).

Conceptele materializează rezultatele procesului de abstractizare, prin care se specifică însușirile esențiale, necesare și suficiente pentru a decide apartenența obiectelor la o clasă sau alta iar **instanțele** sînt noțiuni individuale, particularizări ale conceptelor cărora li s-a dat descrierea. Pe plan mai general, cunoașterea despre cunoaștere și despre reprezentările ei este numită **metacunoaștere**, o cunoaștere de ordin superior, extinsă asupra domeniului de competență al cunoașterii.

Totalitatea pieselor de cunoaștere despre un domeniu, memorate în baza de cunoștințe, alcătuiește un model al lumii la care programul inteligent are acces prin intermediul procedurilor de organizare, clasificare, căutare și recunoaștere. În inteligența artificială, toate aceste componente, piese de cunoaștere plus procedurile de acces la cunoaștere formează ceea ce se numește **sistem cognitiv**. Pentru sistemele cognitive interesează în mod deosebit clasificarea cunoașterii.

2.3.3. Raționament, inferență și euristică

Cunoașterea este o măsură obiectivă a realității, dar prin ea însăși nu reprezintă decît un potențial. În inteligența artificială prezintă interes cunoașterea în acțiune, în care pe baza raționamentelor și judecăților se obțin piese de cunoaștere noi.

Raționamentul este un lanț de judecăți desfășurate în scopul obținerii unor adevăruri noi pe baza cunoașterii disponibile.

Judecata este formula logică fundamentală, cu valoare de adevăr, exprimată printr-o propoziție, în care se afirmă sau se neagă ceva despre altceva. În cazul raționamentului o judecată numită **premisă** este legată de o altă judecată numită **concluzie** prin operația de derivare logică numită **inferență**. Raționamentele care folosesc lanțurile inferențiale pentru trecerea de la premise la concluzii sînt raționamente formale. Ele se aplică situațiilor de completitudine faptică (la care există toate entitățile necesare în rețeaua inferențială). În cazul incompletitudinii faptice, pentru a nu se ajunge la impas în soluționarea problemei a fost dezvoltat

raţionamentul prin analogie sau metaforic.

Inferenţa este deci operaţia prin care, pe baza cunoaşterii unui număr finit de enunţuri numite premise, se trece la acceptarea unui alt element numit concluzie. Într-o inferenţă, premisele formează condiţia suficientă a concluziei, dar aceeaşi concluzie poate fi obţinută şi din alte premise.

O inferenţă este corectă dacă enunţurile componente sînt adevărate şi s-au respectat regulile de formare ale limbajului ştiinţific din teoria în cauză. Schema generală a inferenţei se prezintă în fig. 2.4.



Figura 2.4. Schema generală a inferenţei

Raţionamentele formale sînt de trei tipuri: deductiv, transductiv şi inductiv.

Un raţionament este deductiv dacă premisa constă dintr-o judecată cu caracter general iar concluzia este o judecată cu caracter particular. Caracterul general sau particular se referă la poziţia de subordonare a celor două judecăţi în ierarhia pieselor de cunoaştere.

Raţionamentul este transductiv dacă atît premisa cît şi concluzia se află la acelaşi nivel de generalitate.

Raţionamentul este inductiv dacă produce o concluzie cu caracter general plecînd de la premise cu caracter particular.

Raţionamentul nu foloseşte obiecte fizice, din lumea reală, ci simboluri prin care acestea se reprezintă ca piese de cunoaştere în baza de cunoştinţe. Este vorba de acea abstractizare a realităţii prin care unui obiect, fapt, situaţie, eveniment sau proces i se atribuie un simbol care devine caracteristică. Astfel, cînd observăm realitatea universului contabil, raportăm informaţia la conceptele elementare de mijloace, resurse, activ, pasiv, debit, credit ş.a. care se notează cu simbolurile M, R, A, P, D, C. Cu ajutorul acestora definim concepte contabile noi, rezultate din combinarea conceptelor elementare după legi contabile, folosindu-se apoi în raţionamente cu entităţi de complexitate superioară: cont, balanţă, bilanţ.

Raţionamentele pentru rezolvarea problemelor simbolice se bazează în întregime pe **reguli de inferenţă** - acele procedee prin care se determină consecinţele unor anumite presupuneri ce se fac despre formule sau enunţuri într-o teorie ştiinţifică. Se utilizează în mod frecvent în inteligenţa artificială, procedeul pur formal prin care se deduc teoreme din

axiome prin "înlănţuirea înainte" a inferenţelor sau prin care se determină validitatea unor teoreme enunţate prin "înlănţuirea înapoi" a inferenţelor. Aceste strategii de înlănţuire a inferenţelor se aplică pentru explorarea caracteristicelor sintactice ale reprezentării pieselor de cunoaştere.

Atunci cînd se doreşte soluţionarea problemelor cu luarea în seamă a caracteristicilor semantice ale pieselor de cunoaştere, se foloseşte construcţia formală numită **model**. Orice model are caracteristice regulile de corespondenţă şi semantica. Este vorba de concepte (termeni, semne, simboluri), postulate (axiome şi legi), reguli de transformare a conceptelor şi regulile de corespondenţă între conceptele modelului şi realitatea modelată.

Modelele teoretice ale inteligenţei artificiale sînt cunoscute în limbajul logicii matematice[1]. În stadiul actual al dezvoltării inteligenţei artificiale se cunosc mai bine raţionamentele directe, de tipul celor menţionate anterior sub denumirea de raţionamente formale.

Se cunosc şi raţionamente informale, care pornesc de la semnificaţiile factuale sau procedurale cuprinse în enunţurile problemelor şi surprind interpretări structurale, semnificaţii de natură relaţională, proprietăţi primitive sau complexe, cu ajutorul simbolurilor predicative dependente de termeni care iau valori în mulţimi de tipuri, de structuri ierarhice bazate pe primitive semantice. Raţionamentul informal se bazează şi el pe regulile de inferenţă general valabile.

Euristica este o piesă de cunoaştere menită să sugereze efectuarea de acţiuni plauzibile sau evitarea unor acţiuni neplauzibile, cu rol important în creşterea eficienţei rezolvării problemelor. Termenul de euristică desemnează acea cunoaştere de natură neformală, cu caracter de raţionament bazat pe experienţă, specializare, generalizare şi analogie, cu ajutorul căreia se efectuează o interpretare, se ia o decizie sau se acţionează pentru atingerea unui obiectiv. Se observă că euristica introduce raţionamentul bazat pe reguli care nu provin din formalismul logicii matematice de rezolvare a problemelor, ci din experienţa şi măiestria specifică domeniului problemei.

---

[1]Maliţa, M., „Bazele inteligenţei artificiale", Vol.1, Editura tehnică, Bucureşti, 1987; Georgescu, I., idem, p.22

2.3.4. Problemă, strategie de control şi căutare

Dintr-un punct de vedere mai pragmatic, inteligenţa artificială este privită ca domeniu de cercetare avînd drept obiectiv descoperirea şi utilizarea unor metode foarte generale de rezolvare a problemelor. Deci, rezolvarea problemelor constituie obiectivul central al oricărui sistem de inteligenţă artificială. Componentele sistemului de inteligenţă artificială, care concură la rezolvarea problemelor poartă numele de **sistem rezolutiv**.

Problema se referă la o dificultate de natură cognitivă, o întrebare ce se constituie ca moment iniţial al activităţii inteligente. Dificultatea rezultă din insuficienţa sau inadecvarea răspunsului la întrebarea privind o entitate necunoscută, ori din incapacitatea sistemului rezolutiv de a construi un răspuns prin aplicarea procedeelor de care dispune.

Enunţul problemei are următoarele **componente structurale**: datele, condiţiile şi necunoscutele.

- Datele problemei semnifică enunţurile descriptive referitoare la obiectele abstracte sau concrete, sau cele referitoare la proprietăţile obiectelor.
- Condiţiile problemei sînt specificate prin enunţuri explicite sau implicite despre operaţiile permise a fi aplicate asupra obiectelor menţionate ori referitoare la regulile cărora li se supun operaţiile (aspectul procedural).
- Necunoscutele problemei se referă la obiectele, proprietăţile sau condiţiile neformulate explicit în enunţul problemei, despre a căror existenţă se ştie sau nu că reprezintă o consecinţă a enunţului.

Prin rezolvarea problemei, se va putea da sau nu o determinare fiecărei necunoscute. În procesul rezolvării problemei există iniţial delimitate corect datele şi condiţiile formulate în enunţ. Acestora li se poate adăuga noi piese de cunoaştere din baza de cunoştiinţe a sistemului rezolutiv, căutate şi selectate, fie prin referire directă la enunţ, fie prin lanţuri inferenţiale, ale căror premise au fost formulate în enunţ.

Analiza enunţurilor problemelor care satisfac aceste condiţii de formulare, ne relevă trei mari **categorii de probleme**: probleme bine formulate, probleme incomplet formulate şi probleme greşit formulate.

- La **problemele bine formulate** sînt satisfăcute condiţiile de suficienţă şi necesitate, pentru toate componentele din enunţ iar necunoscutele specificate alcătuiesc împreună cu datele problemei un model consistent;
- În cazul **problemelor incomplet formulate** se disting unele lipsuri, atît în datele problemei, în condiţiile acesteia cît şi în specificarea necunoscutelor;
- La **problemele greşit formulate** se disting contradicţii, inconsistenţe sau componentele problemei nu sînt prezentate clar în enunţ.

A) **Problemele bine formulate** pot fi la rîndul lor: probleme de tip interogativ, predicativ şi imperativ.

- **Problemele interogative** au definite clar trei componente: ipoteza sau datele problemei, procesul la care sînt supuse datele şi rezultatul. Două dintre aceste componente trebuie definite complet, oferind astfel posibilitatea introducerii necunoscutei în cea de a treia componentă. Dacă notăm cu I - ipoteza, cu P - procesul, cu R - rezultatul şi cu X - necunoscuta, atunci putem identifica tipurile de probleme interogative cu ajutorul unei mnemonice în funcţie de poziţia pe care o ocupă necunoscuta între componentele problemei. Astfel, deosebim probleme interogative de tip IPX, de tip IXR şi de tip XPR. În aceste condiţii, dacă toate componentele sînt cunoscute, adică mnemonica este IPR, atunci enunţul problemei reprezintă un fapt. Un fapt devine problemă dacă una dintre cele trei componente este considerată necunoscută.

Problema de tip IPX are enunţul de forma "Care este rezultatul X al aplicării procesului P asupra obiectelor din ipoteza I?" De aici reiese clar că sînt cunoscute ipoteza şi procesul, fie din enunţ, fie din completarea cu piese de cunoaştere din baza de cunoştinţe. Rezultatul X se va determina efectiv prin aplicarea procesului P asupra datelor din ipoteza I. Este vorba de o problemă clasică de prelucrare a datelor.

Problema interogativă de tip IXR corespunde unui enunţ de forma "Ce proces X trebuie aplicat asupra obiectelor din ipoteza I, pentru a obţine rezultatul R?". Este una dintre cele mai delicate probleme ale inteligenţei artificiale, datorită faptului că are caracter creativ. Soluţia sa este abordabilă pe trei căi:

- pe baza unor reguli euristice de forma:

  DACĂ ((ipoteza este de tip t(I) ŞI
  (rezultatul este de tip t(R) ATUNCI
  (metoda este (METODA-1) sau (METODA-n));

- pe baza unor raţionamente prin analogie;
- pe baza unor metode de sinteză procedurală.

Problemelor interogative de tip XPR le corespunde enunţul de forma "Ce obiecte X trebuie prelucrate de către procesul P, pentru a obţine rezultatul R?". Se înţelege că, asemenea probleme pot fi soluţionate direct dacă procesul P admite o inversă notată P^{-1}. Deoarece enunţul problemei ne dă $R = P(X)$, atunci soluţia va fi $X = P^{-1}(R)$. Dacă problema nu

admite o inversă pentru P, atunci numai abordarea prin metode reductive ar putea înlesni obținerea unei soluții.

Problemele predicative au o structură asemănătoare cu acelea interogative, componentele fiind ipoteza, procesul inferențial și concluzia. Natura componentelor diferă, deoarece ipoteza și concluzia sînt entități cu valoare de adevăr (TRUE sau FALSE).

Inducția este problema predicativă în care ipoteza este necunoscută iar rezultatul se obține prin "descoperirea" unor concepte inițiale, a unor cauzalități sau a oricăror premise prin interpretarea rezultatelor obținute pe baza unor reguli de inferență cunoscute. Acest proces rezolutiv care utilizează inducția poartă numele de **învățare din exemple**.

Demonstrația este problema predicativă în care procesul inferențial este necunoscut iar soluția se obține printr-un lanț inferențial, de la ipoteză spre concluzie, care trebuie sesizat ca o concluzie derivabilă din ipoteză. Acest proces rezolutiv este specific demonstrării automate a teoremelor.

Deducția este o problemă predicativă în care concluzia este necunoscută iar soluția se obține aplicînd regulile de inferență asupra expresiilor date prin ipoteză. Deducția este frecvent întîlnită ca urmare a aplicării procedeelor de descompunere a problemelor în subprobleme, și mai rar ca problemă dată de utilizatorii sistemului inteligent.

Problemele de tip imperativ au caracteristic faptul că enunțul lor nu conține necunoscute din triada IPR (ipoteza-proces-rezultat), nu furnizează date de intrare iar rezultatul procesului este apriori cunoscut. Sistemul rezolutiv specific unor astfel de probleme se comportă ca executor al unor comenzi pentru producerea rezultatului cunoscut, adică a efectului actual dorit. Nu este lipsită de sens existența problemelor imperative, dacă avem în vedere posibilitatea ca o situație problematică să fie stratificată pe mai multe niveluri conceptuale, în care o problemă imperativă de pe un anumit nivel să se descompună în probleme de alte tipuri pe nivelurile inferioare.

B) **Problemele incomplet formulate** sînt cele care nu pot avea soluție decît dacă, prin aplicarea unor procedee care nu alterează enunțul, se obține o nouă formă de enunț, care satisface condițiile de bună formulare.

Procedeele care se aplică în astfel de cazuri sînt: reducerea enunțului, completarea enunțului, extinderea obiectivelor și descompunerea problemei.

Reducerea enunțului are loc prin eliminarea elementelor lipsite de

relevanță, care alterează buna formulare a problemei, dar fără a-i afecta specificul problematic.

Completarea enunțului se face cu date transmise prin moștenire de la prototipurile conceptuale existente în alte piese de cunoaștere din enunț sau cu date asumate prin analogie cu obiecte din aceeași categorie, definite mai bine. Se pot folosi și date noi și/ reformulări prin interacțiunea utilizatorului cu sistemul rezolutiv, așa încît să rezulte un nou enunț care satisface cerințele de bună formulare.

Extinderea obiectivelor problemei se referă la o clasă superioară de obiecte care să cuprindă și problema în cauză, numai dacă este posibilă introducerea unor obiecte abstracte, parametri sau condiții suplimentare care să permită reformularea enunțului în vederea satisfacerii condițiilor de bună formulare.

Descompunerea în subprobleme urmărește obținerea unor subprobleme bine formulate și a altora incomplet formulate dar de complexitate mai redusă decît problema inițială, a căror rezolvare să poată fi făcută parțial sau chiar integral, dacă se pot aplica procedeele menționate mai sus pentru buna formulare a problemelor.

Inteligența artificială nu-și propune să rezolve problemele greșit formulate. Totuși, sistemele rezolutive pot fi dotate cu analizoare de probleme care să evidențieze inconsistențele de formulare, furnizînd utilizatorului explicații în legătură cu refuzul sistemului de a soluționa problema.

Orice sistem rezolutiv asigură o succesiune corectă a inferențelor în procesul de rezolvare a problemelor pe baza unei strategii. **Strategia** este însoțită de o procedură de **căutare** pentru selectarea soluției.

Căutarea se concentrează asupra exploatării nodurilor de structuri arborescente sau rețea, celor care descriu în mod frecvent domenii de probleme. Prin contrast, controlul se concentrează asupra explorării metodelor asupra cărora programul de inteligență artificială trebuie să-și îndrepte atenția, cu subprocesele sale inferențiale optime, pentru rezolvarea problemelor.

În fig.nr.2.5 prezentăm direcția de căutare, cu linie punctată, pentru aflarea elementului 8 și controlul în vederea stabilirii metodei inferențiale optime pentru soluționarea problemei.

Strategiile mai importante folosite în controlul raționamentelor sînt: strategia de control înainte, strategia de control înapoi, strategia de control mixtă (înainte-înapoi).

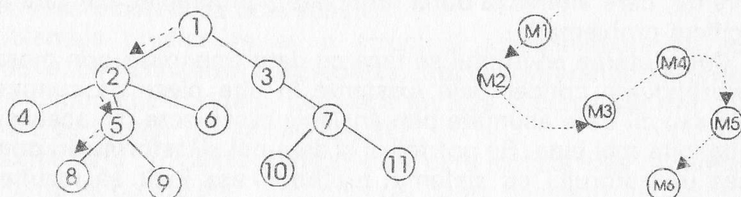


Figura 2.5. Schema pentru control și căutare

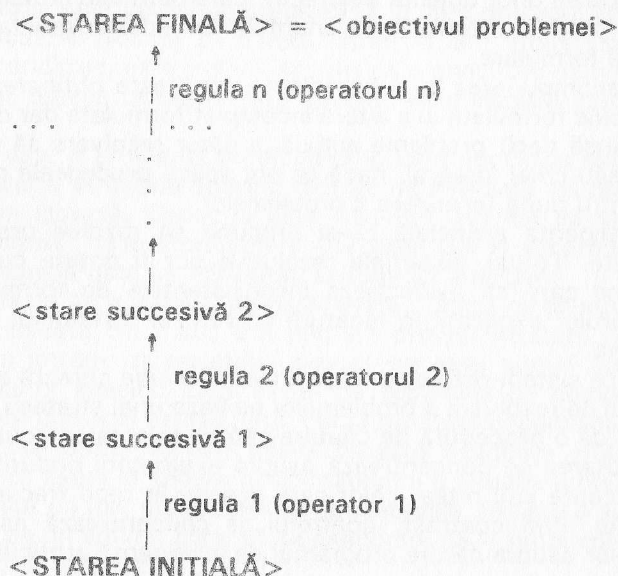


Fig. nr. 2.6. Schema strategiei de control înainte

Strategia de control înainte (forward chaining) se recomandă atunci cînd la problemele de rezolvat sînt prezentate datele sub forma unor proprietăți sau valori asociate unor simboluri, variabile în anumite condiții (stări) și asupra cărora se pot efectua analize, combinări, încadrări în clase sau abstractizări pentru formarea unor noi concepte. Această

strategie este dirijată de datele problemei, de "jos în sus" și determină o căutare în spațiul stărilor. Schema acestei strategii se prezintă în fig 2.6.

În conformitate cu această schemă, se pornește de la starea inițială a faptelor descrise în enunțul problemei și se aplică reguli sau operatori care generează succesiv soluții (stări), pînă cînd se obține răspunsul corespunzător obiectivului problemei. Regulile folosite pot fi reguli de inferență specifice sistemului rezolutiv, reguli definite prin enunțuri referitoare la generarea termenilor, simbolurilor relaționale sau restricțiilor care să reducă spațiul problemei. Operatorii pot să fie și ei exprimați prin reguli. Prin aplicarea regulilor (operatorilor) se obțin date (fapte) noi. La fiecare pas pot fi obținute mai multe fapte valabile, dar numai una se află pe lanțul care duce la soluție. Se obține în acest fel un arbore ale cărui noduri sînt puncte în spațiul problemei iar selectarea soluției obligă la aplicarea unei proceduri de căutare în acest spațiu.

Strategia de control înapoi (backward chaining) se recomandă atunci cînd problemele sînt prezentate spre rezolvare prin enunțarea explicită a obiectivului, care nu se poate obține direct din datele inițiale și necesită folosirea metodelor de reducere la subprobleme. Această strategie de control este deci dirijată de obiectivul problemei sau de "sus în jos", și determină o căutare a subobiectivelor problemei date. Schema acestei strategii o prezentăm în fig.2.7.

Potrivit acestei scheme, se pornește de la obiectivul problemei, urmărind ca prin aplicarea unor reguli să se obțină descompunerea în subprobleme de complexitate mai mică sau reformularea în termeni derivați, pe baza regulii utilizate ajungîndu-se la subobiectivele adecvate.

Strategia de control mixtă (înainte-înapoi) folosește metode de reducere pentru obținerea de subprobleme, pe care le soluționează conform strategiei de control înainte.

2.4. Elemente de bază ale inteligenței artificiale

Componentele inteligenței artificiale pot fi prezentate grafic printr-o schemă cu cadrane (fig.2.8.), la care în centru pe cadrane bine delimitate se află elementele de bază care nu trebuie să lipsească din cultura specialiștilor și din care se constituie, prin creativitate de înalt nivel, domeniile aplica-

ţiilor de pe margine[1].

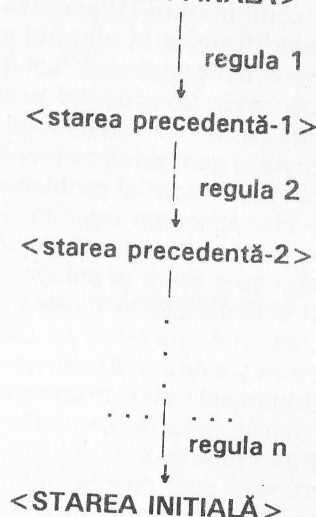


Fig. nr. 2.7. Schema strategiei de control înapoi

Deci, în cadranele interioare identificăm: căutarea euristică, modelarea şi reprezentarea cunoaşterii, raţionamentul inteligent şi logic, limbajele şi instrumentele inteligenţei artificiale. În cadranele exterioare se remarcă celelalte elemente ale inteligenţei artificiale: prelucrarea limbajului natural, rezolvarea problemelor şi planificarea, roboţi inteligenţi şi maşini pentru vedere artificială şi sistemele expert.

2.4.1. Căutarea euristică

Privind cadranul căutării euristice, trebuie să recunoaştem că există metode specializate care introduc în procesul de căutare raţionamente bazate pe reguli care nu provin din formalismul mecanismului de rezolvare obişnuită a problemelor, ci din experienţa acumulată în domeniul problemei.

---

[1]Hunt, D.V., „Artificial intelligence & Expert systems", Sourcebook, Chapman & Hall, New York, 1986, p. 7

Căutarea soluţiilor alternative oferite printr-un raţionament adecvat se realizează pe baza unui arbore ierarhic. Anumite metode specializate sînt proiectate să limiteze spaţiul de căutare folosind informaţii despre natura şi structura problemei din domeniu. Astfel de metode operează adesea prin generarea şi testarea rezultatelor intermediare, pînă la obţinerea soluţiei finale - cea căutată.

2.4.2. Reprezenterea cunoaşterii

Reprezentarea cunoaşterii este problema centrală a sistemelor de inteligenţă artificială. Scopul său îl reprezintă organizarea cunoaşterii cerute într-o anumită formă, la care programul de inteligenţă artificială să poată avea acces pentru luarea deciziei, planificarea, recunoaşterea obiectivelor şi fenomenelor, a situaţiilor, analiza scenelor, elaborarea concluziilor şi execuţia altor funcţii cognitive.

Există specialişti care afirmă clar că problema fundamentală a inteligenţei artificiale o constituie definirea unor metode de reprezentare a unor cît mai mari cantităţi de cunoaştere, sub o formă care să permită folosirea ei efectivă.

Reprezentarea cunoaşterii se poate considera ca o relaţie de definire de forma (1):

$$(1)\ A <\overset{def}{===}> B$$

care stabileşte o legătură între un simbol identificator A, folosit ca nume pentru entitatea de reprezentat şi o expresie B, formulată într-un limbaj de reprezentare, care descrie caracteristicile entităţii şi structura sa în termenii primari, purtători de semnificaţie ai limbajului. Expresia B se numeşte reprezentarea entităţii A sau piesa de cunoaştere A, ea aparţine sistemului cognitiv.

După cum se poate constata, **reprezentarea** constituie o funcţie prin care se preiau fenomene, situaţii, obiecte din realitate şi se transformă în piese de cunoaştere, într-o structură complexă de reprezentare numită bază de cunoştinţe, cu ajutorul unui formalism numit **schemă de reprezentare.** Prezentăm în fig. 2.9. o diagramă instanţială a reprezentării.

În această diagramă se arată cum se ajunge de la realitate la răspuns pe oricare din cele două căi, în care se exemplifică corelaţiile dintre anu-

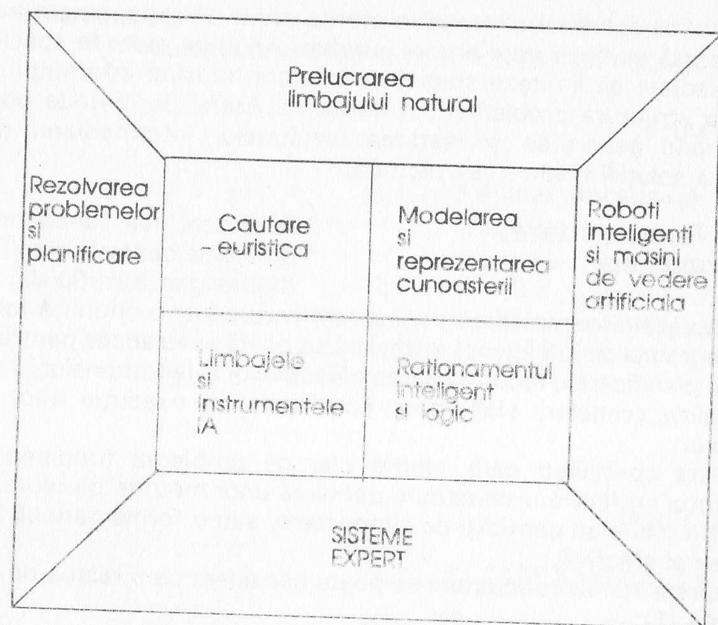


Figura 2.8. Elementele inteligenței artificiale

mite activități care aparțin cunoașterii[1]: observare <---> reprezentare și reprezentare <---> interogare. Diagrama este instanțială, deoarece nu arată cum evoluează realitatea și nici cunoașterea ca urmare a dobîndirii de noi cunoștințe (piese de cunoaștere).

Se înțelege că, din punct de vedere formal, este necesar să se stabilească, în prealabil, modalități corespunzătoare pentru reprezentarea obiectelor, a funcțiilor fiecărei piese de cunoaștere, a relațiilor între piesele de cunoaștere, a regulilor de inferență și a strategiilor de control.

În practică, cunoașterea despre un obiect este dată parțial în termeni descriptivi de proprietăți, parțial în termeni comparativi față de un alt

[1]Călmățuianu, V., Mihăescu, P., Luchian, S., ș.a., Sistemele expert și utilizarea lor în medicină, Raport de cercetare, Universitatea „Al. I. Cuza" Iași, 1986

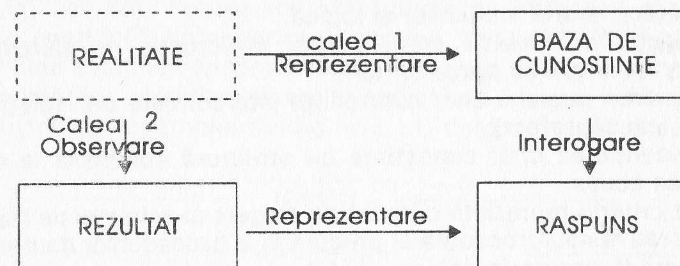


Figura 2.9. Diagrama instanțială (comutativă) a reprezentării cunoașterii

obiect cunoscut, parțial în termeni generativi și parțial în termeni enumerativi.

Întotdeauna este de dorit ca sistemul cognitiv să aibă reprezentări descriptive complete, întrucît acestea sînt formele de bază ale pieselor de cunoaștere din baza de cunoștințe, la care pot fi adăugate și elemente ale altor tipuri de reprezentări folosind raționamentul inferențial adecvat.

Clasificarea și organizarea cunoașterii însoțesc întotdeauna reprezentarea cunoașterii.

În vederea memorării lor în baza de cunoștințe, piesele de cunoaștere sînt clasificate pe baza unor criterii semantice, obținîndu-se clase semnificative sub aspectul relevanței și al genului de proprietăți. În inteligența artificială, clasificarea este orientată către obiectivul aplicației.

Clasificarea cunoașterii este importantă și în explicarea acțiunilor sistemului de inteligență artificială. Astfel, pe plan semantic, în funcție de veridicitatea reflectării realității, cunoașterea poate fi clară sau confuză (vagă, nuanțată), existînd o trecere continuă de la precizie "absolută" la imprecizie "absolută". De aceea, piesei de cunoaștere i se atașează un factor de certitudine, o valoare de adevăr bine nuanțată.

Aceasta reprezintă un aspect dificil al cunoașterii, la care, în cazul judecăților, raționamentelor, implicațiilor cauzale ș.a. asemenea piese de cunoaștere determină folosirea logicii polivalente sau a logicii nuanțate (în care adevărul este marcat printr-o plajă de valori continuă de la fals la adevărat), respectiv obligă la stabilirea plajei de valori pentru nuanțatori lingvistici cum sînt: mai puțin, mai mult, foarte, probabil, credibil, destul de etc., precum și a unor proceduri adecvate pentru combinarea lor.

Tratamentul impreciziei se realizează cu formalisme din teoria probabilităţilor, teoria mulţimilor şi logică.

În funcţie de criteriul complexităţii structurale, în sistemele de inteligenţă artificială se poate urmări:

a) folosirea numai a unor cunoştinţe intercorelate prin relaţii de tip implicaţie (cauză - efect);

b) evidenţierea unor cunoştinţe cu structură complexă la care să putem avea acces.

Acest criteriu reprezintă criteriul de alegere al schemei de reprezentare (declarativă sau procedurală) precum şi al procedurilor de prelucrare a cunoaşterii (funcţionale, logice, obiectuale, etc.).

Pentru primul caz, al cunoştinţelor intercorelate în maniera cauză - efect, se va folosi schema de reprezentare specifică regulilor de producţie. De exemplu:

Regula nr.10:
DACĂ 1) Elementul este de activ SI
2) S-a redus concomitent un alt element de activ
ATUNCI Pasivul rămîne neschimbat

Pentru al doilea caz, al cunoaşterii cu structură mai complexă, se vor utiliza scheme de reprezentare cum sînt reţelele semantice, cadrele şi scenariile. Cu titlu de exemplu dăm în figura 2.10. o reţea semantică pentru fraza următoare, exprimată în limbaj natural:

"Şeful a debitat contul 300 cu 150 lei"

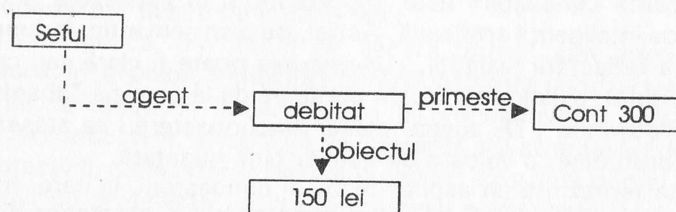


Figura 2.10. Reţea semantică pentru reprezentarea limbajului natural

Gradul de abstractizare al cunoaşterii implică organizarea sistematică a pieselor de cunoaştere printr-o "compactare" a lor şi utilizarea unor

proprietăţi, relaţii şi proceduri cu caracter general, dar cu particularizare rapidă la un caz dat. Pentru această situaţie se utilizează aşa-numita "moştenire ierarhică" (inheritance hierarchy), care constă în obţinerea unor informaţii specifice despre nodurile inferioare din reţea prin cunoaşterea proprietăţilor specifice nodurilor superioare (despre care se spune că sînt moştenite). Exemplul din fig.nr.2.11. demonstrează acest lucru.

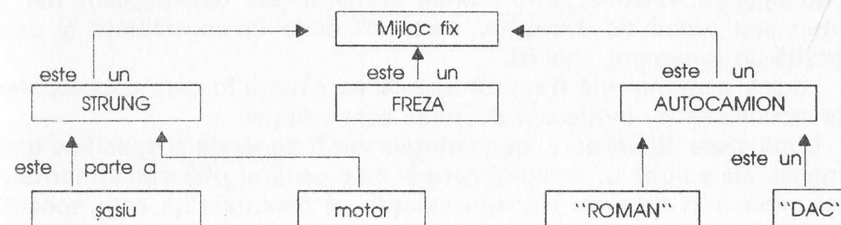


Figura 2.11 Reţea semantică cu moştenire ierarhică

Din fig.2.11. rezultă că, în reţeaua semantică "Mijloc Fix" sînt memorate o singură dată "şasiu" şi "motor" la nivel de strung, fără să mai fie necesară repetarea acestora la celelalte tipuri de mijloace fixe. "Este un" şi "este-parte-a" sînt relaţiile între nivelele ierarhice ale aceluiaşi obiect.

Reţelele semantice sînt folosite cu succes la reprezentarea cunoaşterii în domeniile în care se utilizează taxonomii bine stabilite pentru simplificarea problemei: cercetarea limbajului natural ş.a.

Cadrele se referă la o metodă specială de reprezentare a conceptelor şi situaţiilor. Un cadru (frame), este o structură de date pentru reprezentarea unei situaţii stereotipe printr-o serie de informaţii din care, o parte se referă la modul de utilizare a cadrului, altă parte la ce se va întîmpla în viitor, iar o altă parte la acţiunile necesare cînd acţiunile nu sînt confirmate[1].

Un cadru este organizat aproximativ ca reţeaua semantică. De fapt atît reţelele semantice, cît şi cadrele sînt sisteme bazate pe cadre. Un cadru se prezintă ca o reţea de noduri şi relaţii organizate într-o ierarhie în care nodurile superioare reprezintă obiecte sau concepte generale iar

---

[1]Minski, M., A Framework for Representing Knowledge, în "The Psichology of Computer Vision", P. H. Winston (Ed.), McGraw-Hill Book Co., New York, 1975, p. 23

nodurile inferioare reprezintă instanţe (particularizări) ale acestor concepte.

Descrierea se face cu ajutorul atributelor şi a valorilor asociate. Vezi exemplul din fig.nr.2.12.

După gradul de consistenţă sau inconsistenţă, cunoaşterea se poate diferenţia într-o manieră în care se pun în evidenţă reguli sau legi (piese de cunoaştere valabile pentru orice elemente ale domeniului), dar şi abateri sau reguli de excepţie, care pot duce la contradicţii şi care necesită un tratament special.

Logica nemonotonă tratează asemenea situaţii în care cunoaşterea este inconsistentă, indiferent de sursa contradicţiei.

După sfera de utilizare, cunoştinţele pot fi generale şi specifice unui domeniu. De exemplu, limbajul natural este general utilizabil în comunicarea umană în timp ce limbajul ştiinţific al contabilităţii este specific numai acestui domeniu.

Cunoaşterea poate reflecta şi funcţionarea unui sistem, prin stările sale normale sau anormale. În acest caz, pot exista plaje de valori pentru diversele categorii de stări.

Reprezentarea cunoaşterii constituie, deci, un subdomeniu al inteligenţei artificiale care utilizează metode proprii (teoria formelor ş.a.), dar şi metode din matematică, psihologie, medicină, lingvistică etc.

2.4.2.1. Metode de reprezentare a cunoaşterii

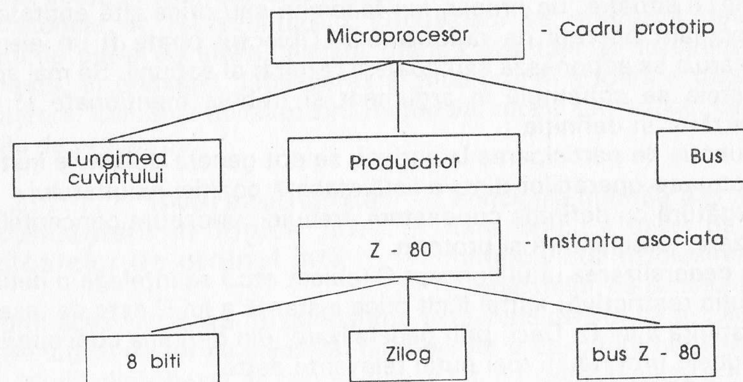
Există trei clase de metode de reprezentare a cunoaşterii: metode logice, metode relaţionale şi metode procedurale.

La **metodele logice**, cunoaşterea se reprezintă sub forma unor enunţuri adevărate (aserţiuni) despre concepte, obiecte, fenomene, situaţii etc. şi relaţiile dintre ele.

Regulile de inferenţă se aplică direct asupra pieselor de cunoaştere din baza de cunoştinţe. Se asigură o cuprindere corectă a semnificaţiei simbolurilor în formalismul utilizat iar reprezentările cunoaşterii sînt simple şi inteligibile.

Soluţiile pentru sistematizarea bazei de cunoştinţe, reprezentarea cunoaşterii despre acţiuni şi a regulilor euristice nu satisfac, motiv pentru care trebuie introduse tehnici suplimentare de organizare şi ataşare a unor proceduri.

În cazul **metodelor relaţionale**, cunoaşterea este reprezentată sub forma grafurilor sau reţelelor, pe baza relaţiilor dintre obiecte sau con-



Microprocesor Z - 80:
Lungimea cuvîntului : 8 biti
Producator : Zilog Co.
Bus-ul : Z- 80

Figura 2.12. Cadru prototip şi instanţa asociată

cepte. Se are în vedere organizarea cunoaşterii în funcţie de omogenitatea sau eterogenitatea domeniului, obţinîndu-se clase şi tipuri variate, conform naturii conceptuale sau factuale a pieselor de cunoaştere, gradului de generalizare sau specializare a conceptelor, tipului relaţiilor şi semnificaţiei.

La **metodele procedurale**, cunoaşterea referitoare la o situaţie evolutivă se reprezintă sub formă de proceduri, care asigură obţinerea unor stări la anumite momente specificate. Cunoaşterea procedurală se invocă prin ataşarea la simboluri, prin reguli de producţie sau pe bază de pattern-uri (şabloane).

În toate metodele menţionate, **reprezentarea** înseamnă definiţia fiecărei piese de cunoaştere din domeniu potrivit formalismului specific metodei. **Definiţia cunoaşterii** înseamnă descrierea formală a unui obiect în termenii limbajului de reprezentare, prin care se diferenţiază obiectele între ele sau se apreciază echivalenţa în cazul obiectelor identice. Prin definiţie se urmăreşte modelarea obiectelor şi a relaţiilor dintre ele, astfel încît să li se specifice toate caracteristicile esenţiale şi să fie permise operaţiile în care obiectele şi relaţiile sînt frecvent implicate.

În acest context, prin obiect înţelegem un obiect fizic, un concept, o acţiune, a situaţie, un proces, un fenomen sau orice altă entitate, cu care efectuăm operaţii de raţionament. Obiectul poate fi un element asupra căruia se acţionează sau poate fi rezultat al acţiunii. Se mai spune că obiectele se constituie în argument şi trebuie menţionate în mod corespunzător în definiţie.

În funcţie de participarea la acţiuni, se pot genera **obiectele instanţă** prin efectuarea operaţiilor dintr-o listă ataşată poziţiei rezultatului.

În legătură cu definiţia cunoaşterii, trebuie cunoscute conceptele de generalizare, specializare şi prototip.

Prin **generalizarea** unui concept C (obiect etc.) se înţelege o definiţie D mai puţin restrictivă, astfel încît orice instanţă a lui C este de asemenea o instanţă a lui D. Deci, prin generalizare, din definiţia conceptului C se omit unele proprietăţi mai puţin relevante astfel:

- dacă PRE(x, D, C) este un predicat în care x este o instanţă a conceptului C şi care, dacă satisface predicatul pentru generalizarea D, atunci va satisface şi definiţia conceptului C:

(1) $DEF(x, D) \wedge PRE(x, D, C) \rightarrow DEF(x, C)$

Cu DEF(x, D) s-a notat predicatul care arată că x satisface definiţia D, cu $\wedge$ conjuncţia, iar cu $\rightarrow$ implicaţia. Orice concept C poate avea mai multe generalizări, iar generalizarea poate fi la rîndul ei generalizată, obţinîndu-se o ierarhie de concepte.

Prin **specializare** se înţelege noţiunea inversă generalizării, astfel:

- definiţia unui concept C este o specializare a definiţiei unui concept D, dacă acesta cuprinde pe lîngă predicatele lui C şi o serie de proprietăţi exprimate prin aceeaşi formulă (1) de mai sus. De exemplu, este posibilă ierarhia de specializări de concepte mai jos:

(2)
$$DEF(x, F) \wedge PRE(x, F, G) \rightarrow DEF(x, G)$$
$$DEF(x, F) \wedge PRE(x, F, H) \rightarrow DEF(x, H)$$
$$DEF(x, G) \wedge PRE(x, G, I) \rightarrow DEF(x, I)$$
$$DEF(x, G) \wedge PRE(x, G, J) \rightarrow DEF(x, J)$$
$$DEF(x, H) \wedge PRE(x, H, K) \rightarrow DEF(x, K)$$
$$DEF(x, H) \wedge PRE(x, H, L) \rightarrow DEF(x, L)$$

Prototipul se referă la obiectul reprezentativ pentru toate instanţele şi care evidenţiază cel mai bine caracteristicile acestuia, acoperind cît mai

complet setul de exemple disponibile.

Metodele de reprezentare a cunoaşterii se utilizează deseori în conjuncţie unele cu altele, fiecare oferind avantaje şi dezavantaje. Se urmăreşte, în principal, înţelegerea şi modificarea uşoară a pieselor de cunoaştere. Cele mai utilizate sînt metodele: reţele semantice, cadrele şi regulile de producţie.

2.4.2.1.1. Reprezentarea cunoaşterii în logica predicatelor de ordinul întîi

Principala metodă de reprezentare logică a bazei de cunoştinţe constă în folosirea logicii predicatelor de ordinul întîi. Conform acesteia, baza de cunoştinţe este considerată ca o colecţie de formule logice de **formă clauzală**, care permit o descriere parţială a lumii reale. Modificările bazei de cunoştinţe se fac prin adăugiri sau ştergeri de formule logice.

Reprezentările logice sînt uşor de înţeles şi au disponibile seturi de reguli de inferenţă necesare operării asupra formulelor logice.

Principalul dezavantaj constă în aceea că sînt mari consumatoare de memorie.

Reprezentarea unei piese de cunoaştere în logica predicatelor de ordinul întîi necesită două etape de bază:

- reprezentarea propoziţională;
- reprezentarea predicativă.

a) **Reprezentarea propoziţională** constă în descompunerea descrierii piesei de cunoaştere în aserţiuni legate între ele prin conectorii logicii calculului propoziţional (conjuncţia, disjuncţia, implicaţia, negaţia şi echivalenţa).

b) **Reprezentarea predicativă** constă în descompunerea aserţiunilor în componenetele lor (predicate şi obiecte asupra cărora acţionează predicaţia) urmată de reprezentarea în formă clauzală.

Exemplu: considerăm fraza definiţie a contului, pe care o descompunem cu respectarea etapelor de mai sus.

"Contul este un mijloc de calcul contabil care stabileşte soldul iniţial şi modificările succesive ale unui element de activ sau pasiv în cursul unei perioade. El serveşte şi pentru efectuarea unor operaţii de calcul contabil care nu privesc direct activul şi pasivul."

După descompunerea în aserţiuni, obţinem:

A1 = Contul este un mijloc de calcul contabil;
A2 = stabilește soldul inițial;
A3 = stabilește modificările succesive ale unui element de activ sau de pasiv în cursul unei perioade;
A4 = efectuează operații de calcul contabil, care nu privesc direct activul și pasivul.

La reprezentarea predicativă se utilizează limbajul calculului cu predicate de ordinul întîi, care ne obligă la exprimarea predicatelor în termenii unor obiecte bine precizate. Astfel, obiectul unic din exemplul nostru este CONT, de fapt un obiect generic, care se instanțiază în obiecte individuale (mulțimea conturilor din planul de conturi: 100, 101, 110,..., 833). În acest caz putem spune că, a fi cont este o caracteristică a fiecăruia dintre obiectele instanță "100", "101", "200" etc., pe care o reprezentăm prin pedicatul CONT(x). Acest predicat este adevărat pentru oricare cont individual.

De exemplu, CONT(100), CONT(200) ș.a.m.d. sînt predicate, deoarece prin definiția generală s-a arătat că sînt conturi.

Trecînd, în continuare, la exprimarea în forma clauzală, se obține reprezentarea:

$$\forall x \ \text{ESTE}(x, A1) \wedge \text{ESTE}(x, A2) \wedge \text{ESTE}(x, A3) \wedge \text{ESTE}(x, A4) \rightarrow \text{CONT}(x)$$

în care: $\forall$ este cuantificator universal;
$\wedge$ conector pentru conjuncție;
$\rightarrow$ este conector pentru implicație.

Reprezentarea clauzală este cel mai simplu și cel mai unitar format pentru toate componentele cunoașterii: obiecte, funcții, relații și reguli de inferență. Deasemenea se pot reprezenta fapte (enunțuri care au întotdeauna aceeași valoare de adevăr).

De exemplu:

→DEBITEAZĂ("300")
→CREDITEAZĂ("700") } pentru asertarea adevărului,

SFOARĂ (MIJLOC FIX)→
STRUNG (MATERIE PRIMĂ)→ } pentru asertarea falsului.

Reprezentarea cunoașterii în formă clauzală corespunde unei metode moderne de elaborare a programelor în inteligența artificială, numită **programare logică** și folosită intens de către limbajul PROLOG și versiunile sale. Metoda are la bază stilul introdus de Kovalski (1979), care folosește predicate logice pentru un set de clauze declarative, în care fiecare clauză are forma:

consecință: - antecedent-1, antecedent-2,..., antecedent-n

în care antecedentele sînt predicate care vor fi testate pentru valoarea lor de adevăr, iar consecința este un predicat, care este TRUE dacă toate antecedentele sînt TRUE.
Un program logic are un **scop** pe care-l compară (matching) cu consecințele memorate în clauze. Cînd găsește o potrivire încearcă să probeze scopul prin considerarea consecințelor împerecheate ca subscopuri. Cînd toate subscopurile sînt TRUE, atunci scopul va fi TRUE. Această căutare prin antecedente pentru a proba consecințele are loc după strategia de control înapoi.

2.4.2.1.2. Reprezentarea cunoașterii prin rețele semantice.

Termenul de rețea semantică introdus de către M.Ross Quilliam, în1966, se folosește pentru a descrie o metodă de reprezentare a cunoașterii bazată pe structura de rețea.

O rețea semantică descrie proprietățile și relațiile dintre obiecte, evenimente, fapte, concepte, situații și acțiuni printr-un graf care constă din noduri și arce etichetate, care le conectează.

Nodurile, într-o rețea semantică, stabilesc obiecte, concepte, evenimente, fapte situații etc.. Arcele definesc relațiile dintre noduri și pot fi reprezentate într-o varietate de moduri, în funcție de felurile cunoașterii de reprezentat.

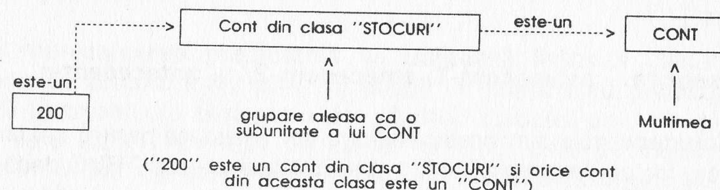
Arcele comune folosite pentru reprezentarea ierarhiilor includ relațiile numite **primitive taxonomice**: "este-un" și "este-parte-a"[1]. Aceste primitive stabilesc proprietatea de "moștenire ierarhică" în rețea, fapt

---

[1]În engleză "IS-A" pentru "este-un" și "HAS-PART" sau "Part-OF" pentru "este-parte-a"

care înseamnă că elementele de pe nodurile inferioare ale rețelei pot moșteni proprietățile elementelor aflate pe nivelurile superioare. Această proprietate determină o economie de memorie deoarece informația despre nodurile similare din rețea nu mai trebuie repetată la fiecare nod.

De exemplu, rețeaua semantică de mai jos:



Schema din acest exemplu ne arată că orice cont din clasa "STOCURI" moștenește toate proprietățile conceptului generic de la nivelul mulțimii CONT.

Pot exista rețele semantice simple, rețele semantice sortate și rețele semantice extinse.

a) **Rețelele semantice simple** au caracteristic faptul că nodurile reprezintă entități individuale, iar arcele reprezintă relațiile dintre acestea. Ele nu introduc specializări ale arcelor și nodurilor, proprietățile și interpretările lor fiind determinate de conținutul descrierii asociate fiecărui simbol care a fost folosit ca etichetă de nod sau ca etichetă de arc.

Cu ajutorul lor se reprezintă orice fel de piesă de cunoaștere. Candidații la poziția de nod inițial se pot selecta după criteriile:

- dacă un nod este etichetat cu un simbol folosit și pentru numele piesei de cunoaștere, atunci acesta va fi nodul inițial al reprezentării;
- dacă este un nod cu proprietatea de a fi nod sursă în toate relațiile în care este implicat, atunci acesta va fi nod inițial.

Descrierea unei relații semantice simple se realizează prin descompunerea sa în triplete de forma:

(⟨nod-sursă⟩ ⟨relație⟩ ⟨nod-detinație⟩)

Astfel de triplete se descriu foarte ușor în limbajul de programare LISP. De exemplu, pentru rețeaua semantică din fig. 2.11:

((ȘASIU ESTE-PARTE-A STRUNG)
(MOTOR ESTE-PARTE-A STRUNG)
(STRUNG ESTE-UN MIJLOC-FIX)

(FREZA ESTE-UN MIJLOC-FIX)
(ROMAN-DIESEL ESTE-UN AUTOCAMION)
(AUTOCAMION ESTE-UN MIJLOC-FIX)
(DAC ESTE-UN MIJLOC-FIX))

Rețeaua semantică se poate reprezenta și prin asocierea la fiecare nod sursă a unei mulțimi de perechi de forma:

(⟨relație⟩ ⟨nod-destinație⟩)

Aceasta reprezintă o modalitate foarte practică de construire a conceptului pentru fiecare nod sursă. De exemplu, rețeaua semantică din fig. 2.13 se reprezintă în LISP astfel:

```
((C7      (ESTE-UN BLOC)
          (ESTE-ÎN GALATA))
 (GALATA  (ESTE-UN CARTIER)
 (ION     (ESTE-UN ADULT)
          (LOCUIEȘTE C7))
 (MARIA   (ESTE-UN ADULT))
 (ADULT   (ESTE-O PERSOANĂ))
 (COPIL   (ESTE-O PERSOANĂ))
 (DIANA   (ESTE-UN COPIL)))
```

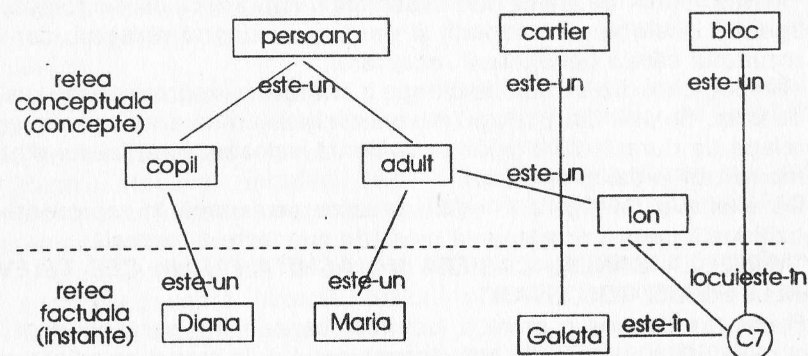


Figura 2.13 **Rețea semantică**

Gruparea relaţiilor în jurul nodurilor sursă permite o bună orientare către metodele de raţionament deductiv.

b) **Reţelele semantice sortate** sînt de mai multe tipuri şi se caracterizează prin folosirea specializării nodurilor şi arcelor, în scopul relevării diferenţelor de natură structurală ale entităţilor reprezentate, diferenţe care nu se pot evidenţia numai prin etichetare.

Specializarea nodurilor constă în crearea unor noduri de tip propoziţie şi completarea cu relaţii cazuale pentru specificarea profunzimilor semantice. Este vorba de a privi propoziţia ca pe un eveniment produs de instanţierea predicatului care indică acţiunea (verbul). Celelalte componente ale propoziţiei fiind termeni ai predicatului, poziţia lor se numeşte caz şi se identifică prin eticheta relaţiei cazuale asociate. Principalele relaţii cazuale sînt:

- predicaţia, care stabileşte că nodul sursă, de tip propoziţional este o instanţiere a predicatului specificat de eticheta nodului destinaţie. Ea poate reprezenta o acţiune sau o stare, fiind echivalentă cu verbul propoziţiei;
- agentul sau actorul, care reprezintă subiectul logic al propoziţiei şi are ca nod destinaţie o entitate cu capacitatea de a efectua acţiunea;
- receptorul, care are ca nod destinaţie o entitate care beneficiază de rezultatul acţiunii;
- obiectul, care are ca nod destinaţie o entitate afectată de acţiunea specificată;
- instrumentul, ce are ca nod destinaţie o entitate pe care o foloseşte agentul la efectuarea acţiunii şi pe care o suportă obiectul, dar de rezultatul căreia beneficiază receptorul;
- timpul, care are ca nod destinaţie o entitate ce reprezintă intervalul de timp, fie prin două relaţii (moment iniţial şi moment final), fie prin relaţia de durată, care pune în evidenţă valoarea intervalului dintre momentul iniţial şi cel final.

De exemplu, în fig. 2.14 dăm o reţea semantică în reprezentare propoziţională pentru următoarea piesă de cunoaştere factuală:

"MÎINE, LA BANCĂ, CASIERA VA ACHITA CU UN CEC TELEVIZOARELE FIRMEI GOLDSTAR"

Predicaţia descrie o stare a lucrurilor care, avînd corespondent în lumea reală, dă o valoare de adevăr predicatului. În raport cu trăsăturile de dinamism şi control, pot exista mai multe **sorturi de predicaţii**:

- acţiunea, corespunzătoare stării de lucruri dinamice şi controlate;
- procesul, corespunzător unei stări de lucruri dinamice şi necontrolate;
- poziţia, corespunzătoare unei stări de lucruri statice , dar necontrolate.

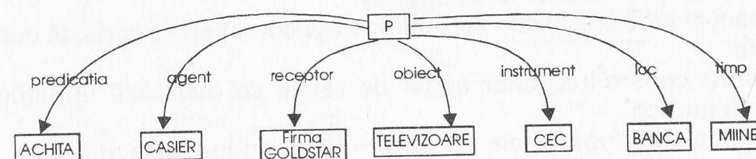


Figura 2.14. Reţea semantică sortată

De exemplu, în fig. 2.15 prezentăm o reţea semantică pentru o piesă de cunoaştere factuală, cu predicaţie din sortul proces:

"INFLAŢIA DIMINUEAZĂ PUTEREA DE CUMPĂRARE A BANILOR"

Întrucît sortul proces este necontrolat, nu există relaţie cazuală de tip AGENT. Singurele relaţii cazuale de bază sînt OBIECTUL şi RECEPTORUL. FORŢA este o relaţie auxiliară, care produce procesul.

Teoria reprezentării cunoaşterii ne relevă şi alte sorturi ale predicaţiei, toate în legătură cu reprezentarea cunoaşterii factuale prin reţele semantice.

Prezintă un interes deosebit şi reprezentarea cunoaşterii conceptuale, în care sînt implicate obiecte abstracte de tipul nodurilor de variabile, al nodurilor de conectori logici şi relaţia de cauzalitate.

Pentru stabilirea ierarhiei între diferitele componente ale reţelei semantice este necesară organizarea. Organizarea cea mai simplă împarte o reţea semantică în:

- reţea conceptuală, în care se descriu piese de cunoaştere reprezentînd concepte, obiecte, acţiuni, procese, fenomene sau stări de lucruri cu caracter generic;
- reţea factuală, în care se descriu instanţe ale conceptelor.

Datorită relaţiei taxonomice dintre concepte şi instanţe, instanţele moştenesc proprietăţile conceptelor. Pot exista şi relaţii suplimentare între componentele reţelei factuale care descriu stări de lucruri particulare

și care nu pot fi avute în vedere la nivelul conceptelor.

c) **Rețelele semantice extinse** se caracterizează prin specializarea nodurilor și arcelor în conformitate cu cerințele programării logice, astfel:

- pentru noduri pot exista sorturile: simboluri constante, simboluri variabile și simboluri funcționale;
- pentru arce pot exista sorturile: relații condiționale și relații concluzive.

Pentru construirea unor astfel de rețele se utilizează următoarele convenții grafice:

- simbolurile funcționale se reprezintă cu noduri hexagonale;
- relațiile concluzive se reprezintă cu săgeți conturate;
- relațiile condiționale se reprezintă prin săgeți pline;
- etichetele arcelor sînt nume de relații și simboluri de predicate.

În figura nr. 2.16 prezentăm o rețea semantică extinsă pentru fraza:

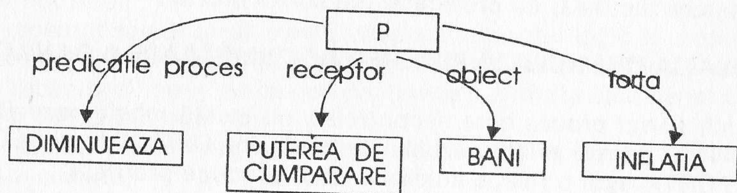


Figura 2.15. Rețea semantică cu predicație din sortul proces

"CINE STUDIAZĂ TEMEINIC AJUNGE UN BUN ECONOMIST"

Exprimarea sub formă clauzală a acestei fraze se prezintă astfel:

AJUNGE(x, BUN-ECONOMIST) ← ESTE(x, STUDENT), ÎNVAȚĂ(x, TEMEINIC)

Din această reprezentare (fig. 2.16) rezultă funcțiile conectorilor logici: conjuncția și implicația. Prezentăm mai jos și funcțiile celorlalți

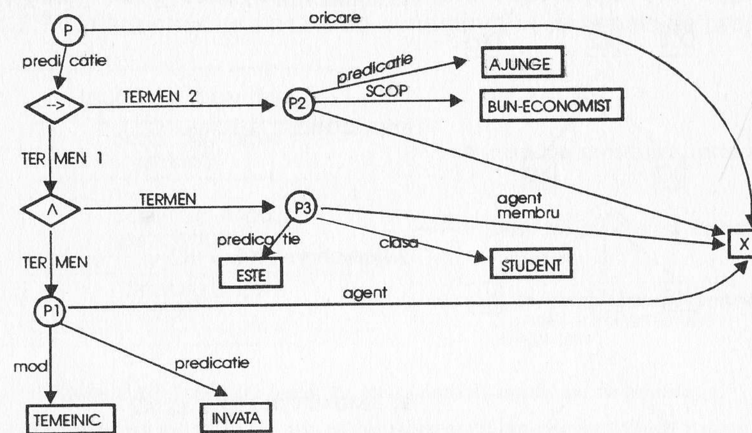


Figura 2.16 Rețea semantică multisortată (cuantificare universală a variabilei)

conectori logici: disjuncția, echivalența și negația.

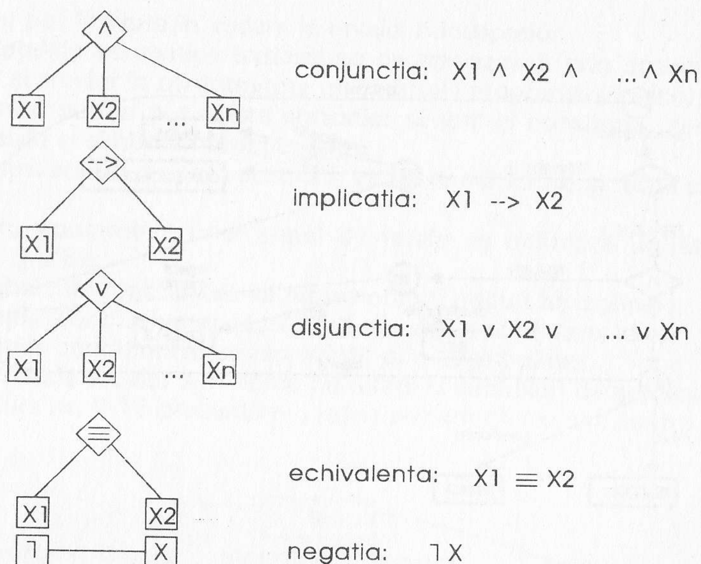
Rețelele semantice extinse sînt de fapt rețele sortate pe baza criteriilor programării logice. Există în literatura de specialitate o mare varietate de rețele semantice extinse: hiperrețele, rețele partiționate ș.a. În toate cazurile rețelele semantice se diferențiază între ele prin:

- primitivele semantice și modalitățile de tratare a neprimitivelor;
- criteriile de stabilire a sorturilor pentru noduri și arce;
- existența mecanismului de definire a unor noi sorturi (noi piese de metacunoaștere);
- convenția de reprezentare grafică.

Rețelele semantice extinse oferă posibilitatea unei reprezentări mai apropiate de realitate și sînt foarte utilizate în inteligența artificială.

2.4.2.1.3. Reprezentarea cunoașterii prin reguli de producție

Reprezentarea cunoașterii cu ajutorul regulilor de producție face parte din categoria metodelor procedurale, fiind cunos-



cută sub numele de sisteme de producție. Metoda regulilor de producție își are originea în teoria limbajelor formale, unde este cunoscută sub numele de reguli de rescriere (rewriting rules).

Mecanismul regulilor de producție a fost propus de către E.Post (1943), în prezent fiind complet pus la punct prin contribuția unui mare număr de cercetători. A.Newell și H.A.Simon au fost primii care prin lucrările lor le-au folosit în aplicațiile de inteligență artificială.

Potrivit acestei metode, organizarea cunoașterii se realizează în trei mari categorii: declarativă (factuală), procedurală și strategică (de control).

Cunoașterea declarativă (factuală) reprezintă piese de cunoaștere sub forma unor structuri de date memorate nerestrictiv într-o colecție denumită **context** sau **bază de fapte**.

Cunoașterea procedurală (operatorie) reprezintă piesele de cunoaștere sub forma unor perechi **premisă-concluzie** sau **condiție-acțiune**, denumite **reguli de producție**, memorate într-o colecție denumită **bază de reguli**.

Cunoașterea strategică (strategia de control) este reprezentată sub

forma unor reguli care privesc desfășurarea procesului de rezolvare a problemelor ca o suită de decizii asupra secvențelor de acțiuni. În fig.nr.2.17. dăm structura bazei de cunoștințe în sistemele de producție.

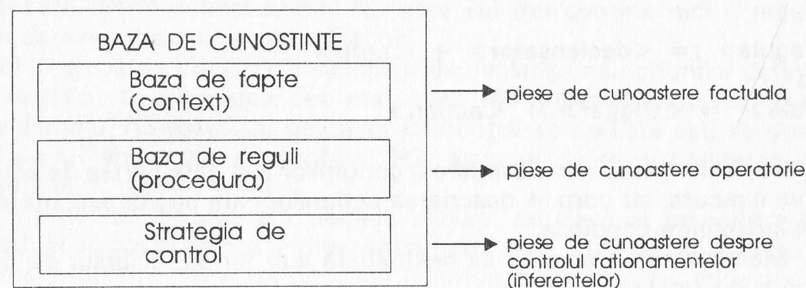


Figura 2.17. Structura bazei de cunoștințe în sistemele de producție.

Un sistem de producție are la bază regula de producție și mecanismul inferențial.

a) **Regula de producție** este o metodă de reprezentare a cunoașterii sub forma:

IF premisă THEN concluzie

sau

IF condiție THEN acțiune

a cărei interpretare se realizează astfel:

DACĂ <este realizată premisa> ATUNCI <se trage concluzia>

sau

DACĂ <partea de condiție este îndeplinită> ATUNCI <se execută partea de acțiune>

După cum lesne se observă, partea din regulă care conține **premisa** sau **condiția** precizează criteriile care trebuie îndeplinite ca să se tragă o concluzie sau să se desfășoare o acțiune. Această parte se mai numește **declanșator** (trigger).

Cînd condiția este îndeplinită se spune despre regula în cauză că este **selectată pentru declanșare**. O regulă selectată pentru declanșare intră împreună cu alte reguli selectate în mecanismul inferențial asupra bazei de fapte (contextului), într-o competiție de stabilire a priorității celei mai mari numită **filtrare**, în urma căreia o singură regulă va fi acceptată

pentru declanşare efectivă, prin execuţia părţii care conţine concluzia sau acţiunea. Partea de concluzie sau acţiune se mai numeşte **corpul regulii.** Din acest motiv în unele lucrări, regula de producţie este prezentată sub forma:

<regula> := <declanşator> + <corp>
sau
<rule> := <trigger> + <action>

în care declanşator = o descriere a condiţiilor sub care partea de acţiune se va executa, iar corp = descrierea acţiunilor care se vor executa dacă sînt îndeplinite condiţiile.

Mecanismul inferenţial se desfăşoară sub forma **ciclurilor de bază,** fiecare în următoarele etape:

- etapa de EVALUARE;
- etapa de EXECUŢIE.

În etapa de EVALUARE (vezi fig.nr.2.18) se determină dacă există în baza de reguli disponibilă, regulile de declanşat şi dacă acestea au partea de condiţie îndeplinită.

În etapa de EXECUŢIE se declanşează regulile reţinute în etapa de evaluare.

Mecanismul poate fi oprit fie în etapa de evaluare fie în etapa de execuţie. Oprirea în etapa de evaluare este determinată de absenţa regulilor declanşatoare iar oprirea în etapa de execuţie poate fi determinată de un ordin dat de către o regulă declanşată.

a) Etapa de EVALUARE cuprinde în principiu trei faze: SELECŢIA sau RESTRICŢIA, FILTRAREA şi REZOLVAREA CONFLICTELOR[1].

SELECŢIA sau RESTRICŢIA determină, plecînd de la starea curentă sau trecută a bazei de fapte şi a bazei de reguli, un subansamblu F1 al bazei de fapte şi un subansamblu R1 al bazei de reguli, care pot fi comparate în timpul fazei de filtrare care urmează (vezi fig.nr.2.18).

FILTRAREA (pattern matching, în engleză) realizează compararea părţii declanşator din fiecare regulă din subansamblul R1 cu subansamblul F1 de fapte. Toate regulile care sînt compatibile cu faptele din F1 (a căror condiţii au fost satisfăcute) sînt grupate într-un subansamblu R2 de

---

[1]Farreny, H., „Les systèmès experts. Principes et exemples", p. 45 ş.u. Vezi şi Buchanan, B.G., ş.a., Rules Based Expert Systems, p. 50 ş.u.

reguli, numit "ansamblu de conflict". Regulile acestui subansamblu intră într-o competiţie în urma căreia numai o regulă va fi declanşată.

REZOLVAREA CONFLICTELOR. În timpul acestei faze se determină un subansamblu de reguli R3 dintre regulile din R2 care vor fi efectiv declanşate. Dacă subansamblul R3 este vid (nu conţine nici o regulă) etapa de execuţie nu va mai avea loc.

b) În etapa de EXECUŢIE se comandă declanşarea acţiunilor definite prin regulile cu prioritatea cea mai mare din subansamblul R3. Dacă subansamblul R3 este vid, procesul inferenţial se opreşte sau se poate relua de la "ansamblul de conflict" R2, examinîndu-se posibilitatea declanşării altor reguli din R2.

În practică, fiecare din etapele ciclului de bază al mecanismului inferenţial se pot prezenta în diverse variante şi se înţelege, că soluţionarea unei probleme oarecare necesită înlănţuirea automată a milioane de cicluri de bază. Fiecare asemenea ciclu de bază presupune execuţia a mii sau chiar milioane de instrucţiuni. În cazul calculatoarelor electronice aflate în uz curent ne putem aştepta la viteze de cel mult cîteva mii de cicluri inferenţiale pe secundă. Calculatoarele din generaţia a 5-a pot realiza viteze de ordinul miliardelor de LIPS (Logical Inference Per Second).

Din mecanismul inferenţial rezultă o caracteristică importantă a sistemelor de producţie - comunicarea între reguli se realizează prin intermediul bazei de fapte.

Consecinţa acestei caracteristici o constituie faptul că, prin partea de acţiune a regulii de producţie se urmăreşte realizarea comunicării prin introducerea sau scoaterea unei piese de cunoaştere din baza de fapte. Ca urmare, prezintă interes deosebit **corectitudinea specificării regulilor de producţie** prin exprimarea adecvată a condiţiilor şi acţiunilor.

Am văzut că forma în care apare partea de condiţie în formularea obişnuită a regulii de producţie presupune explorarea bazei de fapte pentru a găsi piese de cunoaştere care corespund clauzelor din care este alcătuită condiţia.

De exemplu: Specificarea narativă a regulii de producţie.

DACĂ furnizorul oferă produsul-A cu preţul-P1;
ŞI furnizorul oferă produsul-B cu preţul-P2;
ŞI produsul-B este produs nou:
ATUNCI acceptă produs-B;
ŞI trimite delegat;

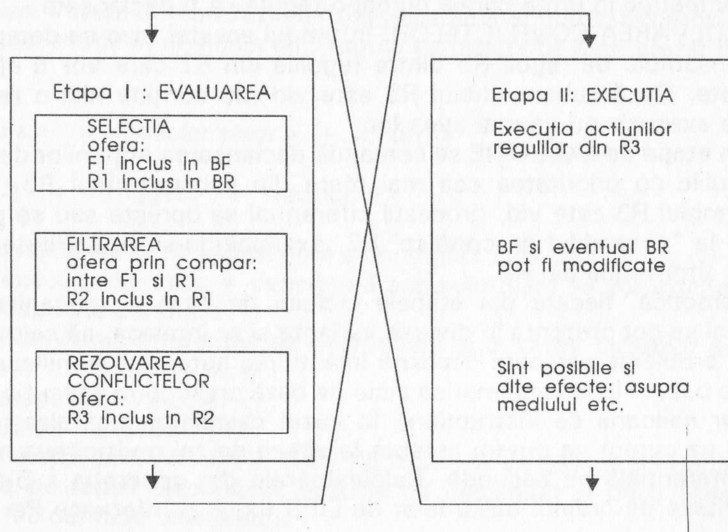


Figura 2.18. Schema ciclului de bază al mecanismului inferențial (BF = baza de fapte, BR = baza de reguli)

ȘI plătește produs-B.

Forma în care este introdusă regula în baza de reguli depinde de tipul compilatorului sau interpretorului limbajului de programare ales.

De exemplu, dacă se folosesc exprimări factuale, se va utiliza limbajul LISP astfel:

(BF (FURNIZORUL OFERĂ PRODUSUL-A CU PREȚUL-P1)
(FURNIZORUL OFERĂ PRODUSUL-B CU PREȚUL-P2)
(PRODUSUL-B ESTE PRODUS NOU))

unde BF = baza de fapte.

În această situație și regulile se vor exprima tot în limbajul LISP, astfel:

(REGULA-10)

(CONDIȚIE (FURNIZORUL OFERĂ PRODUSUL-A CU PREȚUL-P1)
(FURNIZORUL OFERĂ PRODUSUL-B CU PREȚUL-P2)
(PRODUSUL-B ESTE PRODUS NOU))
(ACȚIUNE (ACCEPTĂ PRODUSUL-B)
(TRIMITE DELEGAT)
(PLĂTEȘTE PRODUSUL-B))))

După cum se poate observa, faptele din BF satisfac condiția pentru declanșarea regulei-10. Avem deci, de a face cu un **mecanism de corespondență simplă**. La rezolvarea problemelor reale acest mecanism nu permite tratarea cunoașterii incomplete. De aceea este necesară folosirea variabilelor împreună cu mecanismul de interpretare corespunzător, pentru atribuirea de valori și legarea de acestea a simbolurilor din condiții cît și din acțiuni, în toate aparițiile următoare. Mai mult, folosirea variabilelor se leagă cu cea a pattern-urilor (șabloanelor). De exemplu:

pentru șablonul (A B ?X D)
și baza de fapte BF = {(A B C D),
(E B C D),
(A B B A),
(A B B D)}

există două corespondențe ale șablonului în baza de fapte, și anume:
(A B C D) care leagă variabila X la valoarea C;
și (A B B D) care leagă variabila X la valoarea B.
Cu ?X s-a marcat variabila X, care are semnificația de cuantificare universală.

Utilizarea variabilelor oferă posibilitatea folosirii regulilor de producție în care, în locul unor piese de cunoaștere instanțializate, să se introducă concepte. Astfel, prin legarea variabilelor pe întreaga specificație a regulii, se pot construi instanțe diferite ale aceleași reguli, dacă în baza de fapte există fapte care corespund șablonului din condiție.

De exemplu, considerăm regula de moștenire a proprietăților:

(REGULA -MP
(CONDIȚIE (?X APARȚINE CLASA ?Y)
(?Y ARE PROPRIETĂȚILE >Z))

(ACȚIUNE (?X ARE PROPRIETĂȚILE >Z)))

și baza de fapte:

((CONT-DE-CHELTUIELI APARȚINE CLASA CONT)
(CONT ARE PROPRIETĂȚILE
(ESTE MIJLOC DE CALCUL CONTABIL)
(STABILEȘTE SOLDUL INIȚIAL)
(STABILEȘTE RULAJUL ÎN CURSUL PERIOADEI)
(STABILEȘTE SOLDUL FINAL)))

Conform acestor descrieri, condiția regulii are corespondență în baza de fapte deoarece prima clauză se satisface prin legarea variabilei ?X la simbolul CONT-DE-CHELTUIELI și legarea variabilei ?Y la simbolul CONT. Prin propagarea legării în următoarele clauze, cea de-a doua clauză devine

(CONT ARE PROPRIETĂȚILE >Z),

care are corespondența în baza de fapte și leagă variabila de tip segment >Z la segmentul definit în cea de a doua piesă de cunoaștere ș.a.m.d., încît regula menționată mai sus va deveni o instanță, astfel:

(REGULA-MP1
(CONDIȚIE (CONT-DE-CHELTUIELI APARȚINE CLASA CONT)
(CONT ARE PROPRIETĂȚILE
(ESTE MIJLOC DE CALCUL CONTABIL)
(STABILEȘTE SOLDUL INIȚIAL)
(STABILEȘTE RULAJUL ÎN CURSUL PERIOADEI)
(STABILEȘTE SOLDUL FINAL)))
(ACȚIUNE (CONT-DE-CHELTUIELI ARE PROPRIETĂȚILE
(ESTE MIJLOC DE CALCUL CONTABIL)
(STABILEȘTE SOLDUL INIȚIAL)
(STABILEȘTE RULAJUL ÎN CURSUL PERIOADEI)
(STABILEȘTE SOLDUL FINAL))))

Acțiunea din REGUL-MP1 va determina (produce) introducerea unei instanțe ca nouă piesă de cunoaștere în baza de fapte. Este vorba de partea de acțiune a acesteia (REGULA-MP1).

În șabloane, în afara simbolurilor de variabile prefixate cu ? sau cu > se mai pot utiliza simbolurile prefixate cu < și *, astfel:

<X - cînd valoarea variabilei participă la procesul de intrepretare a regulii;

*X - cînd variabilei i s-a atribuit ca valoare un alt simbol și interesează valoarea acestuia;

?X - pentru o variabilă individuală;

>X - pentru variabila de tip segment (ca în exemplul de mai sus).

S-a observat că **specificarea acțiunii** în regulile de producție respectă cerințele limbajului de programare ales, uneori sub forma de funcții specifice limbajului, funcții definite de utilizator.

Partea de acțiune (corpul regulii) poate preciza operații asupra bazei de fapte și operații de activare sau dezactivare a unor reguli, de inițiere a execuției unor programe sau de modificare a conținutului bazei de reguli.

Operațiile asupra bazei de fapte sînt:
- introducerea unei piese de cunoaștere în baza de fapte;
- scoaterea unei piese de cunoaștere din baza de fapte, operație care poate avea efect și asupra informației auxiliare privind rezolvarea conflictelor;
- modificarea unei piese de cunoaștere prin schimbarea și înlocuirea vechii piese cu o nouă piesă în aceeași poziție.

Operațiile asupra bazei de reguli sînt:
- introducerea, scoaterea sau modificarea unor reguli de producție;
- activarea sau dezactivarea unor reguli de producție.

Operațiile de inițiere a execuției unor programe sînt cele care:
- efectuează calcule pentru atribuirea de valori unor simboluri (subrutine în alte limbaje);
- efectuează operații cu dispozitivele periferice;
- modifică strategia de control, servind unor procese decizionale diferite de cele implementate prin programul de inteligență artificială în limbajul ales.

În aceste condiții, folosirea metodei regulilor de producție impune construirea unor piese de cunoaștere cu structuri adecvate bazei de fapte și bazei de reguli, pornind de la elementele extrase din acestea sau obținute ca urmare a unor operații specificate anterior, construirea unor liste cu parametri formali pentru programele inițiate în execuție, respectiv construirea de contexte locale ca memorie de lucru pentru regulile de executat.

Metoda regulilor de producție se bucură de un mare succes datorită

simplităţii regulilor, modularităţii, formatului liber al pieselor de cunoaştere şi naturaleţei transpunerii în reguli a cunoaşterii despre starea şi comportarea sistemelor în lumea reală. Sistemele de producţie se bucură de o autonomie ridicată, întrucît regulile pot fi tratate ca piese de cunoaştere independente, fără a fi necesară analiza efectului lor asupra altor reguli.

Pe măsura acumulării cunoaşterii, sistemele de producţie îşi diminuează performanţele dacă se schimbă tipul problemei de rezolvat. Un alt factor al diminuării eficienţei îl constituie faptul că execuţia acţiunilor are loc după etapele menţionate anterior pentru ciclul mecanismului inferenţial.

Cel mai bine, regulile de producţie corespund aplicaţiilor din domeniile în care cunoaşterea este factuală iar secvenţele de acţiuni nu pot fi predeterminate, ori în procesele care pot fi descrise prin acţiuni independente, fără a fi necesară şi prestabilirea unui flux de control.

Cauzele care duc la diminuarea performanţelor sistemelor de producţie pot fi eliminate prin măsuri de organizare a bazei de fapte, a bazei de reguli şi introducerea unor elemente auxiliare care să accelereze ciclul inferenţial.

Organizarea bazei de fapte are în vedere cuprinderea numai a pieselor de cunoaştere care sînt implicate direct în rezolvarea unei probleme concrete. De aceea, se procedează la gruparea pieselor de cunoaştere în funcţie de criterii de natură taxonomică. Deci, cunoaşterea trebuie să fie adecvată tipului problemei. Gruparea pieselor de cunoaştere pe tipuri (categorii) de probleme se realizează prin **învăţare din experienţa sistemului**:

- pentru fiecare piesă de cunoaştere se păstrează înregistrarea referinţelor către problemele în care este implicată.

Aceleaşi criterii se folosesc şi pentru **organizarea bazei de reguli**, dar se ţine seama în mod suplimentar şi de **specializarea funcţională** a regulilor, pe grupuri care se realizează în funcţie de similitudinea obiectivelor urmărite prin aplicarea regulilor de producţie individuale.

Organizarea presupune şi adăugarea unor **contexte** în care, sistemul înregistrează şi menţine mereu actuală informaţia necesară accelerării procesului de căutare. Se urmăreşte restrîngerea căutării numai la acele entităţi ale bazei de cunoştinţe care pot candida la încheierea cu succes a unei etape a ciclului inferenţial.

Există metode de reprezentare noi, care includ aşa-numitele **contexte de referinţe** prelucrate de un **filtru**, o componentă a interpretorului specific limbajului ales.

În continuare, prezentăm **formulele metalingvistice** pentru sintaxa generală a regulilor de producţie:

<regula-de-producţie> ::=
<identificator-regulă> <şablon-contextual> <specificaţie> |
<regula-de-producţie> <specificaţie>
<identificator-regulă> ::= <nume> | <tip> <nume>
<tip> ::= <înainte> | <înapoi> | <control>
<înainte> ::= F-
<înapoi> ::= B-
<control> ::= C-

Formulele pentru regulile strategiei de control înainte (F-reguli):

<şablon-contextual> ::= <stare>
<specificaţie> :: <acţiune>

Formulele pentru regulile strategiei de control înapoi (B-reguli):

<şablon-contextual> ::= <concluzie>
<specificaţie> ::= <premisă>

Formulele pentru regulile de tipul <control> (C-reguli):

<şablon-contextual> ::= <stare>
<specificaţie> ::= <identificator-regulă>

Regulile de producţie de tipul <control> se aplică acelor reguli pentru care informaţia de <tip>, care le ataşează unei strategii de control predefinite, nu a fost specificată sau care servesc pentru a decide tipul de strategie care se aplică într-o anumită situaţie.

2.4.2.1.4. Reprezentarea cunoaşterii prin cadre.

În domeniul inteligenţei artificiale termenul de cadru (frame) se referă la o metodă specială de reprezentare a conceptelor, obiectelor, fenomenelor, situaţiilor etc. Acest

termen este propus de Marvin Minsky în 1974[1]. Pe ideile lui M.Minsky s-au elaborat limbaje de reprezentare a cunoaşterii orientate pe concepte, ca AIMDS, FRL, KRL, KL-ONE, SRL şi Units.

Un cadru este un format (pentru reprezentarea unui concept, obiect sau situaţie stereotipă), alcătuit din următoarele elemente sintactice:

- nume, folosit ca identificator simbolic pentru obiect sau concept etc.:
- rubrici (slot), pentru diverse categorii de atribute specifice conceptului/obiectului/situaţiei;
- faţete, perechi de forma simbol-valoare (vezi exemplul de mai jos).

Exemplu de descriere a unui cadru pentru microcalculatorul Felix PC:

Cadru:: FELIX PC	— nume cadru
Rubrica:: CPU	— nume rubrică
Faţeta 1:: I-8086	— simbol şi valoare (faţetă)
Faţeta 2:: I-80286	
Rubrica:: Main Memory	
Faţeta 1:: RAM-1	
Faţeta 2:: ROM	
Faţeta 3:: PROM	
Rubrica:: Peripheral Devices	
Faţeta 1:: Floppy Disk A	
Faţeta 2:: Floppy Disk B	
Faţeta 3:: Line Printer	

[1]Minsky, M., Ibidem, p. 25 ş.u.

Faţeta 4:: Console
Rubrica:: Preţ
Faţeta 1:: 350000
Faţeta 2:: 550000

Potrivit acestei piese de cunoaştere, din exemplul de mai sus, orice microcalculator Felix PC aparţine clasei microcalculatoarelor. Din punct de vedere structural, Felix PC este un microcalculator profesional ale cărui atribute sînt trecute în fiecare rubrică.

De fapt, conceptul de cadru se referă la acea structură de rubrici libere în care, pe timpul prelucrării, se vor plasa simbolurile şi valorile purtătoare de semnificaţie care definesc piesa de cunoaştere în mod concret. Această operaţie de plasare a simbolurilor şi valorilor în timpul prelucrării se numeşte **umplere sau fill-in** (în engleză).

Cadrele au o structură arborescentă surprinsă cel mai bine în definiţia metalingvistică specifică limbajului de reprezentare FRL (Frame Representation Language), şi anume:

<cadru> ::= (<nume-cadru> <descriere-rubrici>)
<descriere-rubrici> ::= (<nume-rubrică> <descriere-faţete>)
(<descriere-rubrici>(<nume-rubrică> <descriere-faţete>)
<descriere-faţete> ::= (<nume-faţetă> <descriere-date>)|
(<descriere-faţete>(<nume-faţetă> <descriere-date>)
<descriere-date> ::= (<valoare>) | <descriere-date>(<valoare>)
<valoare> ::= <date> | <date>(<eticheta> <mesaje>)|
<date>(<eticheta> <mesaje> <cometarii>)

Se poate observa cu uşurinţă că în exemplul anterior am utilizat o descriere liberă de forma:

CADRU::<simbol>
RUBRICA::<simbol>
FAŢETA::<valoare>

Simbolurile pot să reprezinte proprietăţi (atribute) ale obiectelor, conceptelor, dar şi nume ale altor cadre sau de proceduri ataşate,

obţinîndu-se astfel o reţea semantică potrivită unei situaţii particulare. Din acest motiv, unii autori definesc noţiunea de cadru ca pe o reţea de noduri şi relaţii organizate într-o ierarhie, unde nodurile superioare reprezintă concepte generale, iar nodurile inferioare reprezintă instanţe ale acestor concepte.

Descrierea efectivă a unei piese de cunoaştere prin cadre înseamnă folosirea unui limbaj de reprezentare specific pentru exprimarea rubricilor şi faţetelor, în conformitate cu specificaţia simbolurilor.

Cadrele pot include şi proceduri care permit descrierea acţiunilor de umplere a rubricilor şi faţetelor referite în descrierea piesei de cunoaştere. De exemplu:

Cadru:: SALARIAT
Rubrica:: SALAR
Faţeta:: COMUTAŢIE

VALOARE:: (PLUS AVANS SPOR COMPENSAŢIE ALOCAŢIE REST-DE-PLATĂ)

În acest exemplu s-a folosit o procedură de **COMUTAŢIE** care va umple simbolul SALAR cu VALOARE calculată din cele cinci componente ale salariului, în conformitate cu specificaţiile limbajului LISP. Cînd este întîlnit simbolul SALAR pentru o instanţă de SALARIAT (nume concret de persoană), atunci se va comuta pe procedura ataşată de simbolul VALOARE, calculîndu-se salariul cuvenit persoanei.

În afara **procedurilor de comutaţie** pot exista:

- procedurile numite **capcane (trap,** în engl.) ataşate numai instanţelor, pentru a introduce o anumită specificitate pe o piesă de cunoaştere concretă;

- procedurile numite **demoni**, ataşate implicit, prin simpla lor invocare (ca urmare a apariţiei unei situaţii specifice, a unei schimbări de stare, apariţiei unui eveniment etc.) şi nu prin descriere explicită.

O **procedură demon** poate supraveghea situaţiile tipice legate de procesul de interpretare a cadrelor, astfel:

-demonul IF-ADDED, supraveghează aducerea unui fapt în baza de cunoştinţe care corespunde unui şablon de invocare dat;

- demonul IF-REMOVED supraveghează ştergerea unui fapt din baza de cunoştinţe care corespunde unui şablon de invocare dat;

- demonul IF-NEEDED supraveghează apariţia în procesul de interpretare a unei condiţii de invocare corespunzătoare unui şablon dat.

Pentru **implementarea demonilor**, limbajele de inteligenţă artificială dispun de funcţii adecvate. De exemplu, limbajul LISP permite implementarea unui demon printr-o funcţie definită de programator conform structurii următoare:

<demon> ::= (DEFINIŢIE-DEMON <nume> <tip> <şablon> <corp>
<tip> ::= IF-ADDED | IF-REMOVED | IF-NEEDED

în care **<şablon>** este o expresie de corespondenţă (matching) care poate include şi variabile; **<corp>** este programul LISP care va fi interpretat automat la apariţia situaţiilor de invocare implicită a procedurii. De exemplu, pentru transferul unei persoane de la un compartiment la altul, în cadrul aceleiaşi întreprinderi, se poate folosi definiţia de demon de mai jos:

```
DEFINIŢIE-DEMON TRANSFER
        IF-NEEDED
(?X TRANSFERAT DE LA COMPART ?Y LA COMPART ?Z)
    (PROG (REMOVE        (CADRU:: ?Y (RUBRICA::PERSOANA
          (FAŢETA::?X))))
          (ADD           (CADRU::?Z (RUBRICA::PERSOANA
          (FAŢETA::?X))))))
```

Piesele de cunoaştere despre acţiuni se pot reprezenta prin **cadre şi scenarii**, astfel încît în afara componentelor declarative, cadrul să conţină şi o componentă de tip procedural.

Conceptul de scenariu (în engl. **script**) este introdus de R.C. Schank şi P.R. Abelson[1] în legătură cu cel de cadru, şi se referă la sistematizarea folosirii componentelor cadrului în vederea cuprinderii atît a informaţiei declarative cît şi a celei interpretative (procedurale), astfel:

- **partea declarativă** a scenariului conţine rubrici de tip cadru pentru

---

[1]Schank, R.C., Abelson, R.P., Script, plans, goals and understanding, Lawrence Erlbaum Associattes, 1977

obiecte, locul de desfăşurare a evenimentelor, timpul şi modalităţile de desfăşurare;

- **partea procedurală** conţine rubrici pentru acţiunile de efectuat sub formă de secvenţe de program pentru evenimentele normale, pentru lipsa unor obiecte, pentru situaţia de eroare sau alte defecţiuni.

Partea procedurală este urmată de specificarea evenimentului de început şi a evenimentului care constituie obiectivul principal. Iată mai jos un model de scenariu pentru activitatea în cadrul laboratorului de "Sisteme expert în contabilitate."

Scenariul de mai jos poate fi îmbunătăţit şi cu alte situaţii care să prevadă evenimente nedorite împreună cu modul de acţiune pentru soluţionarea lor. În acest scop pot fi instalaţi demoni care să trateze astfel de situaţii (lipsa studenţilor, lipsa cadrului didactic, defectarea echipamentelor etc.).

În mod normal, programul de inteligenţă artificială care va interpreta cunoaşterea din scenarii trebuie să poată face raţionamente ipotetice. Pot exista scenarii în care atît agentul acţiunii cît şi locul de desfăşurare sînt procesorul care intrepretează piesa de cunoaştere. Există astfel rubrici implicite care se notează cu "*". Rubricile pentru timp şi durată se pot nota cu "***" şi se pot înlocui cu variabile ale căror valori se pot utiliza în aceste scopuri.

Metoda reprezentării cunoaşterii prin cadre se aplică nu numai pentru fapte ci şi pentru regulile bazei de cunoştinţe. Există tot mai multe implementări reuşite în care cadrele s-au folosit pentru reguli. Un exemplu îl constituie limbajul de reprezentare a cunoaşterii RBFS (Rule Based Frame System).

În acest limbaj o **bază de cadre** reprezintă o colecţie de enunţuri despre obiecte sau alte entităţi din lumea reală împreună cu relaţiile care există între ele. Aceste enunţuri au forma de noduri pentru obiecte sau entităţi şi legături etichetate pentru relaţii care în limbaj se transcriu în substantive (nodurile) şi respectiv în verbe (legăturile)[1].

---

[1]Esfahani, L., Kellet, I., Integrated graphical approach to knowledge reprezentation and acquisition, în "Knowledge Based Systems", vol.1, No. 5 Dec/1988, p. 303

SCENARIU:: Laborator de sisteme expert în contabilitate
AGENT:: ASISTENT şi STUDENŢI
LOCUL DESFĂŞURĂRII:: SALA LD2
TIMP:: SĂPTĂMÎNAL 100 MINUTE
PROGRAM:: **EVENIMENT-1:: DESCHIDERE LABORATOR** **EVENIMENT-2:: INTRAREA ÎN SALĂ** **EVENIMENT-3:: PREGĂTIREA MATERIALULUI DI-DACTIC** **EVENIMENT-4:: PREGĂTIREA MICROCALCULATO-RULUI** **EVENIMENT-5:: ENUNŢAREA PROBLEMEI** **EVENIMENT 6:: RĂSPUNSURI LA EVENTUALE ÎNTREBĂRI** **EVENIMENT-7:: REZOLVAREA PROBLEMEI** **EVENIMENT-8:: CONTROLAREA SOLUŢIILOR** **EVENIMENT-9:: DISCUTAREA SOLUŢIILOR** **EVENIMENT-10:: FORMULAREA CONCLUZIEI** **EVENIMENT-11:: OPRIREA MICROCALCULATO-RULUI** **EVENIMENT-12:: COMUNICAREA TEMEI VIITOARE** **EVENIMENT-13:: ÎNCHIDEREA LABORATORULUI** **ÎNCEPUT:: EVENIMENT-1** **OBIECTIV:: EVENIMENT-7**

În inteligenţa artificială, cadrul a fost introdus ca o structură generică de reprezentare a cunoaşterii, dar, în fapt, se poate utiliza în cele mai diverse scopuri. În general, cadrul poate reprezenta un **concept clasă** deoarece poate descrie un set de concepte structural identice numit **cadru prototip**. Cadrul care reprezintă un membru al conceptului clasă, adică un concept individual se numeşte **cadru instanţă**.

Un cadru prototip este o structură de definiţii de tipuri care sînt prezentate de către propriile rubrici. Între acestea, o rubrică poate fi **rubrică terminal** cînd este singulară, ori **rubrică non-terminal** cînd este ea însăşi structurată.

Un cadru instanţă este o instanţiere a unui cadru prototip, identic din punct de vedere structural, care are aceleaşi rubrici, umplute uneori chiar cu valori nule, dacă lipseşte cunoaşterea.

Rubrica terminal conţine o valoare (întreg sau şir). Rubrica non-terminal este umplută cu unul sau mai multe cadre instanţă, care sînt instanţieri ale prototipului şi care definesc tipul rubricii (fig.2.19.).

PROTOTIP	calculator	preţ	CPU	MU	disp. perif.
este un INSTANŢA	calculator personal	preţ	CPU	MU	disp. perif.
este un INSTANŢA	FELIX PC	preţ	CPU	MU	disp. perif.
		550000	I - 8086	RAM	Disc Floppy Consola Imprimantă

Figura 2.19. Relaţia dintre prototip şi instanţă

2.5. Medii de programare pentru inteligenţa artificială

"Inteligenţa artificială fără unul din limbaje este fizică pentru poeţi - lăudabilă şi utilă, dar nu total serioasă."

Patrick H. Winston

Inteligenţa artificială a devenit o ştiinţă experimentală care dezvoltă programe cu un comportament inteligent. În acest scop se utilizează limbaje şi alte instrumente de programare, care simplifică lucrul în timpul construirii sistemelor de inteligenţă artificială - activitate interactivă şi incrementativă.

Aşa cum programele clasice (algoritmice) se dezvoltă treptat, în

etape, cu posibilitatea revenirii la etapele anterioare, folosindu-se medii de programare adecvate, programele de inteligenţă artificială necesită medii de programare interactive, cu mijloace de alocare dinamică a memoriei, pe măsura dezvoltării, exprimarea funcţiilor recursive şi manipularea simbolică.

În prezent, mediile de programare a inteligenţei artificiale se orientează în patru mari direcţii:

- limbaje de programare;
- limbaje de ingineria cunoaşterii;
- instrumente pentru construirea sistemelor;
- instrumente pentru ajutor în programare.

În principiu, orice mediu de programare a inteligenţei artificiale trebuie să aibă următoarele trăsături:

- să fie puternice, mentenabile şi să dispună de limbaje standardizate;
- să posede biblioteci bine dotate cu cod sursă, cod obiect şi cunoştinţe din domeniu;
- să existe video-terminale cu facilităţi grafice de înaltă rezoluţie, color, cu ferestre multiple şi actualizare rapidă;
- să dispună de facilităţi software pentru utilizare uşoară;
- echipamentele de intrare să fie excelente;
- comunicarea interprocese să fie standardizată şi flexibilă;
- software de interfaţă prietenos şi uniform;
- să dispună de editor care să trateze programe bazate pe structura individuală;
- după dezvoltarea pe cele mai bune medii de programare, programul să poată fi translatat în alte limbaje şi maşini mai potrivite ca viteză şi memorie de lucru.

2.5.1. Limbajele de programare

Limbajele de programare folosite în aplicaţiile de inteligenţă artificială sînt:

- limbajele orientate pe problemă (FORTRAN, PASCAL, etc.);
- limbajele orientate pe obiect (SMALLTALK ş.a.);
- limbajele de manipulare a simbolurilor (LISP şi PROLOG).

Din rîndul acestor limbaje, limbajele de manipulare a simbolurilor sînt cele mai propice aplicaţiilor de inteligenţă artificială.

LISP este cel mai vechi şi mai faimos limbaj de programare funcţională.

În anul 1960 John McCarthy de la M.I.T. (S.U.A.) a dezvoltat **LISP**

ca limbaj de prelucrare a listelor (LISt Processing), cu facilităţi de recursivitate pentru descrierea problemelor. În LISP datele şi programele au forma de expresii simbolice (S-expresii), care sînt memorate ca structuri de tip listă. LISP operează cu două tipuri de concepte: **atomi şi liste.**

Atomii sînt simboluri (constante sau variabile) folosite ca identificatori de obiecte. Ei pot fi numerici sau nenumerici (idei, fapte, lucruri ş.a.m.d.).

O listă este o secvenţă de zero sau mai multe elemente incluse între paranteze, în care fiecare element poate fi un atom sau o listă.

LISP are **trei funcţii de bază**, care pot fi utilizate în S-expresii:

- funcţia CAR, care extrage capul listei;

Exemplu: (CAR (ECONOMIST)) are ca rezultat E;

- funcţia CDR, care extrage coada listei;

Exemplu: (CDR (CONTABIL)) are ca rezultat L;

- funcţia CONS, care permite construirea unor noi liste. Ea foloseşte ca argumente două S-expresii şi produce o nouă listă al cărei CAR este primul argument, iar CDR este al doilea argument.

Exemplu: (CONS 'ECONOMIST'(CONTABIL INTELIGENT)) are ca rezultat lista (ECONOMIST CONTABIL INTELIGENT).

Pentru scrierea funcţiilor mai complexe, LISP dispune de alte două **funcţii foarte eficiente:**

- funcţia EQ, cu două argumente atomice, care produce TRUE dacă atomii sînt identici şi FALSE în caz contrar;

Exemplu:

(EQ 'ECONOMIST'INGINER) produce FALSE;

(EQ 'ECONOMIST'ECONOMIST) produce TRUE;

- funcţia ATOM, cu un singur argument, care produce TRUE dacă argumentul său este un atom şi FALSE în caz contrar.

Aceste funcţii se mai numesc **predicate**, deoarece produc valorile T şi F (de la TRUE şi FALSE).

O expresie condiţionată în LISP se scrie în forma:

```
(COND (c1 a1)
      (c2 a2)
      "
      "
      "
      (cn an))
```


unde ci sînt condiţiile iar ai acţiunile.

Acţiunea unei expresii concrete constă în evaluarea condiţiilor în ordine, pînă cînd este întîlnită una care nu produce valoarea **NIL** (lista vidă). Acţiunile corespunzătoare condiţiilor sînt atunci executate.

Cu folosirea acestor posibilităţi este uşor de scris noi funcţii pentru manipularea S-expresiilor. De aici calitatea lui LISP ca limbaj de programare funcţională.

De exemplu, funcţia EQUAL va determina dacă două S-expresii sînt identice:

```
(DEFUN EQUAL (X Y)
(COND ((ATOM X) (COND ((ATOM Y) (EQUAL X Y)
                       (T NIL)))
       ((ATOM Y) NIL)
       ((EQUAL (CAR X) (CAR Y)) (EQUAL (CDR X) (CDR Y)))
       (T NIL)))
```

Notă: Funcţia precizată mai sus, EQ, nu îndeplineşte aceste cerinţe, deoarece este definită pentru argumente atomice. În LISP, foarte criticată este proliferarea parantezelor, care necesită o grijă deosebită în programare.

Din cele relatate pînă aici rezultă că LISP dispune de un mecanism de manipulare a simbolurilor sub forma structurilor de tip listă. Reamintim că lista este o simplă colecţie de entităţi închise între paranteze mici, unde fiecare entitate poate fi reprezentată printr-un simbol. Structurile de tip listă sînt utile pentru reprezentarea conceptelor, obiectelor şi a tuturor situaţiilor complexe pentru care se atribuie simboluri adecvate.

LISP este cel mai popular şi mai utilizat limbaj pentru aplicaţiile de inteligenţă artificială, fiind, alături de PROLOG, cel mai recomandat. Popularitatea LISP-ului derivă din posibilităţile sale:

- uşurinţă şi flexibilitate în manipularea simbolurilor;
- managementul automat al memoriei;
- posedă mijloace complexe de editare şi corectare a programelor;
- tratează uniform textul programului sursă şi baza de cunoştinţe. Această posibilitate permite ca un program LISP să-şi scrie singur programe care învaţă reguli noi sau le modifică pe cele existente.

LISP poate fi procurat în mai multe dialecte, pentru cele mai diverse tipuri de calculatoare, inclusiv sub formă de maşini LISP, special construite pentru dezvoltarea şi execuţia rapidă şi eficientă a programelor. De

exemplu: MACLISP, COMMON-LISP iar în ţara noastră TC-LISP, DM-LISP, LISP-86, tLISP.

În LISP, relaţiile dintre concepte, obiecte etc., sînt caracterizate ca liste conţinînd o relaţie folosită de către obiectele legate între ele. LISP are o mare flexibilitate, dar nu izbuteşte să ghideze reprezentarea cunoaşterii sau a mecanismelor pentru acesarea bazei de cunoştinţe cu scheme de control proprii. Ca urmare strategiile de control trebuie elaborate de către programator.

PROLOG (PROgramming în LOGic) este dezvoltat în 1972 de către A. Colmerauer şi P. Roussel de la Universiatea din Marsilia. Prin natura sa PROLOG este un sistem de demonstrare a teoremelor. Programele PROLOG constau din "axiome" în logica predicatelor de ordinul întîi, împreună cu un scop de atins (teorema de demonstrat). Axiomele sînt scrise în forma clauzelor Horn. De exemplu, regula se scrie astfel:

P IF Q1 AND Q2 AND... Qk pentru care $k>0$;
dacă $k=0$ atunci P este un fapt;
P, Q1...Qk sînt termeni.

Programarea logică se referă la folosirea regulilor şi faptelor în reprezentarea cunoaşterii şi folosirea deducţiei pantru a răspunde la întrebări.

În mod concret, un program PROLOG constă dintr-un grup de proceduri în care partea stîngă a unei proceduri este un pattern (şablon) care poate fi instanţiat pentru a îndeplini scopurile din partea dreaptă a aceleiaşi proceduri. În aceste condiţii programarea în PROLOG presupune specificarea unui set de fapte şi reguli pentru a forma o colecţie de cunoştinţe despre un subiect oarecare. În continuare, PROLOG cu tehnicile sale accesează baza de cunoştinţe şi realizează inferenţele necesare.

Mai exact, programele PROLOG constau dintr-un set de clauze. Clauza poate fi un fapt, o regulă sau o întrebare despre piesele de cunoaştere depozitate în baza de cunoştinţe. Răspunsul la întrebări se realizează prin aplicarea regulilor la setul de fapte din baza de cunoştinţe.

Sintaxa PROLOG este foarte simplă. Faptele sînt exprimate în formă naturală. De exemplu, faptul "contul este de activ" poate fi scris în PROLOG astfel:

(ESTE (CONT, ACTIV)

Relaţia ESTE trebuie urmată de către obiectele puse în relaţie, CONT şi ACTIV, închise între paranteze mici.

Programele PROLOG sînt implementate aşa încît să opereze în manieră conversaţională. Imediat după introducerea faptelor este posibil să se lanseze întrebările. Întrebările sînt prefixate cu "?-". De exemplu, prin introducerea enunţului:

?- **ESTE (CONT, ACTIV)** *PROLOG* va răspunde cu Yes sau No.

PROLOG suportă variabile care se pot utiliza inclusiv pentru a cere răspuns la întrebări generale. De exemplu, la întrebarea:

?- **ESTE (X, ACTIV)** se va putea răspunde cu X=CONT

În aceeaşi manieră se pot specifica şi regulile, cu respectarea formei de mai jos:

predicat (argument$_1$, argument$_2$,..., argument$_n$): -scop

unde semnul :- se interpretează prin IF, iar scop este ceea ce trebuie să se demonstreze, cu folosirea faptelor din baza de cunoştinţe.

PROLOG rezolvă problemele prin compararea şabloanelor (pattern matching), care poate fi privită ca o operaţie de unificare în sensul logicii predicatelor de ordinul întîi. Procesul de rezolvare începe cu căutarea primei clauze (proceduri) a cărei parte din dreapta se potriveşte cu scopul. Procesul de căutare poate fi ghidat de programator prin alegerea ordinei procedurilor, datelor şi scopurilor în clauze.

PROLOG poate fi considerat ca o extensie a lui LISP cuplat cu limbajul de cereri a bazelor de date relaţionale, care utilizează relaţii virtuale. Ca şi LISP, PROLOG este un limbaj de programare interactiv şi foloseşte alocarea dinamică a memoriei. În PROLOG căutarea este controlată de un interpretor, un mecanism care analizează şi procesele şi clauzele.

Cei care critică limbajul PROLOG spun că are un cod obiect ineficient, e mare consumator de timp şi e dificil de bănuit ce se întîmplă în dezvoltările sale. Cei care apără acest limbaj arată că PROLOG prezintă avantajul reprezentării celor mai sofisticate procese inferenţiale, putînd lucra cu baze de date relaţionale şi dispune de compilatoare eficiente. Lucru extraordinar pentru specificul aplicaţiilor din contabilitate.

În prezent PROLOG este implementat pe o mare varietate de calcu-

latoare, inclusiv microcalculatoare. Are şi el multe dialecte. Se pare că cel mai performant este TURBO PROLOG pentru microcalculatoare compatibile IBM PC/XT/AT/PS-2.

Alte limbaje pentru inteligenţă artificială ca: SAIL, QLISP, POP-2, PLANNER, CONNIVER, AMORD au fost create şi dotate cu facilităţi de organizare a cunoaşterii şi căutării. Multe dintre ele însă, nu au mai rezistat concurenţei lui LISP şi PROLOG.

Actualmente, în inteligenţa artificială, limbaje hibride din LISP şi PROLOG (LOGLISP, LISLOG ş.a.), ca şi limbaje din familia celor orientate pe obiecte (SMALLTALK, PLASMA, PARC etc.) suscită un interes crescînd.

Atenţie deosebită prezintă **maşinile de inteligenţă artificială** construite pe un cuvînt de 32 de biţi, mult mai adecvate pentru administrarea memoriilor de mare capacitate, de care programele de inteligenţă artificială au atîta nevoie. Maşinile de inteligenţă artificială au permis abordarea **programării exploratorii** cu limbaje de programare orientate pe obiecte.

În **limbajele de programare orientate pe obiecte** (SMALLTALK, PARC) programatorul poate crea **clase** care descriu **obiecte** şi poate implementa un sistem de inteligenţă artificială prin descrierea **mesajelor** ce se transmit între obiecte.

Un obiect se prezintă sub forma unei entităţi, care este o combinaţie dintre date şi proceduri (fig.2.20).

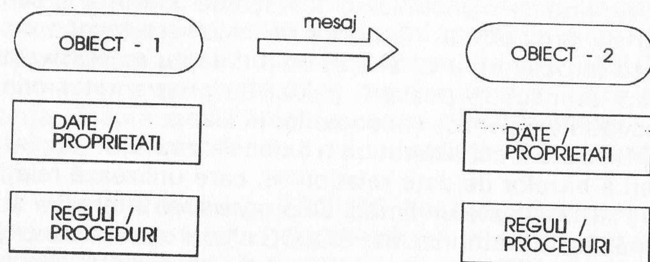


Figura 2.20. Transmiterea mesajelor în programele orientate pe obiect

Vocabularul SMALLTALK dispune de cinci elemente de bază: obiect, clasă, instanţă, metodă şi mesaj. Semnificaţia lor o prezentăm, pe scurt, mai jos:

- Obiect = colecţia de date şi proprietăţi la care sînt ataşate procedurile;

- Clasa = descrierea unui set de obiecte;
- Instanţa = un obiect din clasă;
- Metoda = o procedură care implementează o operaţie;
- Mesaj = un apel de procedură, care cere un obiect să-şi execute una din metodele proprii.

SMALLTALK-80 este limbajul care se bazează pe comunicaţia cu obiecte, astfel: mesajele sînt trimise obiectelor iar obiectele returnează alte obiecte drept răspuns. Puterea acestui limbaj constă în faptul că toate entităţile sînt obiecte şi toate obiectele sînt manipulate într-o manieră consistentă. Un mesaj constă dintr-un receptor, un selector şi argumente. Pentru evaluare, mesajul este trimis receptorului. Selectorul descrie operaţia pe care receptorul cere să o execute. Pot exista mesaje unare şi mesaje binare.

Mesajele unare au nume, dar, nu au argumemte. De exemplu, expresia "debitează cont" trimite mesaj la obiectul CONT cu selectorul "debitează".

Mesajele binare conţin un singur argument şi un selector luat dintr-un set de selectori speciali. De exemplu, mesajul binar "a + 2" are selectorul " + " şi argumentul "2", ambele în legătură cu obiectul "a".

Descrierea unui obiect se numeşte **clasă**. O clasă conţine un set de metode. Fiecare metodă se identifică cu un şablon. Cînd obiectul primeşte un mesaj, se invocă metoda selectată de un selector al viitorului mesaj. Fiecare metodă răspunde pentru manipularea argumentelor care însoţesc mesajul.

Limbajul SMALLTALK-80 este numai o parte a mediului de programare cu acelaşi nume, deoarece în afara interpretorului propriu-zis, mediul mai conţine diverse utilitare ca: editoare, depanatoare, instrumente de dezvoltare (browsers) a programelor şi suport pentru grafică. SMALLTALK-80 este interactiv, portabil şi uşor de utilizat.

2.5.2. Limbaje de ingineria cunoaşterii

Această categorie de limbaje include limbaje create special pentru reprezentarea cunoaşterii, administrarea bazei de cunoştinţe, construirea sistemelor bazate pe reguli, fiind specializate în dezvoltarea sistemelor expert. Ele sînt de fapt instrumente complexe pentru construirea sistemelor de inteligenţă artificială şi constau dintr-un limbaj pentru construirea unui sistem expert, integrat într-un mediu suport extins cu alte utilitare.

Limbajele de ingineria cunoaşterii sînt de două tipuri: limbaje vide

(schelet)și limbaje cu scop general.

Limbajele vide (schelet) numite și shell systems, oferă structura și facilitățile de construire a sistemului expert într-un domeniu concret (de exemplu: EMYCIN shell, adică Empty MYCIN, ExperTeach, ExSys ș.a.). Literatura și practica le numește Expert System Shell, fiind adevărate medii de programare, deoarece conțin mecanisme de inferență prefabricate, interfața pentru dialog și modulele de achiziție a cunoașterii.

Limbajele cu scop general pot fi utilizate în manipularea cunoașterii în cele mai diverse domenii și tipuri de probleme.

Ele prezintă avantajul generalității și flexibilității.

Exemplu:M1, OPS5, OPS83, KES, IBM ESE ș.a.

Limbajele de ingineria cunoașterii poartă denumirea de Knowledge Engineering Language și fac în continuare obiectul unor cercetări asidui.

2.5.3. Instrumente pentru construirea sistemelor

Instrumentele pentru construirea sistemelor sînt programe inteligente elaborate în scopulde a ajuta la achiziția și reprezentarea cunoașterii, respectiv la proiectarea sistemelor bazate pe cunoștințe în vederea construirii lor. Intră în această categorie instrumentele de ajutor în proiectare (AGE) și instrumentele de achiziție a cunoașterii (TEIRESIAS și ROGET).

2.5.4. Instrumente pentru ajutor în programare

Instrumentele pentru ajutor în programare includ utilitare specializate în depanarea programelor de inteligență artificială, editoarele pentru baza de cunoștințe și mecanisme de construire a intrărilor-ieșirilor și explicațiilor. De regulă, aceste instrumente fac parte integrantă din mediile suport pentru sistemele expert. De exemplu: EXPERT-EASE, PLUME, ș.a.

2.6. Aplicațiile inteligenței - artificiale în economie, contabilitate și finanțe

2.6.1. Aplicațiile generale ale inteligenței artificiale

De aproximativ un deceniu sistemele de inteligență artificială au căpătat o remarcabilă dezvoltare în numeroase activități asistate de calculatoare. În legătură cu acest fenomen există două aspecte majore care trebuie relatate:

- domeniile unde asemenea sisteme se aplică sînt din ce în ce mai variate;
- folosirea acestor sisteme are o specificitate extremă.

Limitele potențiale ale aplicațiilor inteligenței artificiale sînt tot atît de vaste încît pot cuprinde virtual întreaga gamă a activităților umane. Pentru a demonstra aceasta este suficient să vizualizăm tabelul aplicațiilor generale de mai jos.

Tabel nr. 2.1. Aplicațiile generale ale inteligenței artificiale

DOMENIUL	APLICAȚIILE
1) Administrarea cunoașterii	Accesul la baze de date inteligente; Achiziția cunoașterii; Înțelegerea textelor; Generarea textelor; Traducerea automată; Explicațiile; Operații logice asupra bazelor de date;
2) Comunicarea interumană	Înțelegerea vorbirii; Generarea vorbirii

DOMENIUL	APLICAȚIILE
3) Învățarea și învățămîntul	Învățarea asistată de calculator; Învățarea asistată de calculatoare inteligente; Învățarea din experiență; Generarea conceptelor; Funcționarea și întreținerea sistemelor de instruire automată;
4) Diagnoza defecțiunilor (erorilor) și depanarea sistemelor	Oameni; Mașini și utilaje; Sisteme de orice tip;
5) Calculul simbolic	Matematici aplicate; Operații cu sisteme "Fuzzy" (sisteme vagi); Programarea automatelor;
6) Comunicații	Accesul publicului la bănci de date via telefon și cu înțelegerea vorbirii; Interfețe în limbaj natural pentru produse informatice;
7) Operații ale mașinilor și sistemelor complexe	Fabricația cu calculator integrat (Computer Integrated Manufacturing). Mecatronica
8) Sisteme inteligente autonome	Vehicule autonome
9) Management	Planificare; Programare; Monitorizare;
10) Întreținerea și interpretarea sensorilor	Captarea semnificației de sensorii de date; Integrarea sensorilor multipli pentru dezvoltarea sistemelor de captare de înaltă performanță;

DOMENIUL	APLICAȚIILE
12) Percepție vizuală și ghidarea automată	Inspectarea; Identificarea; Verificarea; Ghidarea; Vizualizarea; Monitorizarea;
13) Asistarea inteligentă	Diagnoza medicală, financiară, contabilă, tehnică, etc. Sisteme expert interactive; Instrumente de construire a sistemelor expert;
14) Medicina	Diagnoză și tratament; Supraveghere pacient; Mașini de citit pentru orbi; Automatizarea cunoașterii medicale;
15) Știință și inginerie	Descoperirea legilor științifice; Validarea cunoașterii; Sinteze în chimie și biologie; Interpretarea datelor și informațiilor științifice; Ajutoare în proiectarea inteligentă a experimentelor și testelor;
16) Industrie	Managementul producției; Planificarea și programarea producției; Roboți inteligenți; Planificarea proceselor de orice tip; Mașini inteligente; Inspectarea asistată de calculator; Mecatronica;

DOMENIUL	APLICAȚIILE
17) Apărare	Consilieri experți; Interpretarea și sinteza senzorilor; Fotointepretarea automată; Vehicule autonome; Evaluarea situațiilor conflictuale...
18) Internațional	Înțelegerea și interpretarea scopurilor, aspirațiilor și motivelor diferitelor țări și culturi, a modelelor generale; Traducerea limbajului natural.

2.6.2. Activitățile de bază ale sistemelor de inteligență artificială

În cartea "Building Expert Systems", Hayes-Roth[1] arată că activitățile de bază ale sistemelor de inteligență artificială sînt:

- interpretarea situațiilor (deducția unei situații obținute de la sensori speciali);
- predicția (inferarea consecințelor unor situații existente);
- proiectarea (configurarea unor obiecte și sisteme în condiții restrictive);
- planificarea (proiectarea acțiunilor);
- monitorizarea (compararea rezultatelor curente cu cele așteptate și orientarea acțiunilor);
- depanarea (prescrierea de remedieri pentru disfuncționalitățile unui sistem, obiect etc.);
- reparearea (execuția planului de remedieri prescrise pentru obiect sau sistem);
- învățarea (diagnoza și corectarea comportamentului elevilor, studenților și personalului);
- controlul (supravegherea și guvernarea comportamentului sistemelor).

Se observă unele similitudini cu opinia exprimată de D.V.Hunt.

2.6.3. Aplicațiile specifice ale sistemelor de inteligență artificială

Nu lipsite de interes ne apar aplicațiile specifice prezentate de autorul menționat mai înainte în domenii de cea mai mare actualitate (tabelul 2.2.)

Tabelul nr.2.2. Aplicații specifice ale sistemelor de inteligență artificială

DOMENII	APLICAȚII
1) Servicii	Acces la baze de cunoștințe pentru rezervarea locurilor la mijloace de transport; Controlul traficului aerian; Controlul traficului la sol;
2) Financiar-contabil	Pregătirea taxelor și impozitelor, a situațiilor financiar-contabile etc. Sisteme expert; Consultanți inteligenți;
3) Management executiv	Citirea corespondenței și spoturilor publicitare care prezintă interes; Planificare;
4) Administrarea resurselor naturale	Prospecțiuni geologice; Administrarea resurselor prin sensori la distanță;
5) Activități spațiale	Diagnoză și reconfigurare; Planificare și programare; Operații la sol; Operații în spațiu ș.a.

Oricît ar părea de diverse, toate aceste aplicații au legături directe cu activitatea economică, vizînd eficiența în primul rînd și integrarea sub multiple aspecte.

De altfel, conjuncția dintre sistemele economice și inteligența artificială a condus la concentrarea eforturilor asupra introducerii sistemelor de inteligență artificială în domeniul producției (roboții și fabricația asistată CIM, CAM), urmează desfacerea producției (marketing), funcția finaciar-contabilă, planificarea și administrarea.

Credem că este interesant de menţionat mai pe larg aria aplicaţiilor economice, aşa cum se desprinde din bogata literatură de specialitate. Încercăm acest lucru în continuare.

În cadrul sistemului informatic integrat al agentului economic sînt două moduri de utilizare a sistemelor de inteligenţă artificială:

a) în locul unor aplicaţii informatice deja existente. De exemplu, aplicaţia privind selectarea furnizorilor, diagnoza financiară, previziunile contabile etc;

b) înlocuirea în întregime a unui subsistem sau a unei aplicaţii informatice clasice cu un sistem expert adecvat. De exemplu CLASS şi CAPOSS-E (IBM 1983) implementate pentru programarea producţiei care reacţionează mai bine în cazul unor întreruperi a fabricaţiei prin decizii pertinente, sînt utilizate drept consultanţi pentru demonstrarea deciziilor în probleme de organizare a producţiei, putîndu-se adapta dinamic la funcţiile întreprinderii.

Tendinţa o constituie ca sistemul informatic integrat să încorporeze sisteme expert pentru cele mai diverse activităţi. În principiu, recomandările pentru dezvoltarea unor noi sisteme de inteligenţă artificială în cadrul unei întreprinderi se bazează pe o matrice corespunzătoare funcţiilor, ca în tabel nr. 2.21.

După cum se observă, se au în vedere funcţiile şi subfuncţiile în care există deja sisteme expert implementate. Noi am ţinut seama de experienţa mondială deja acumulată[1].

Tabel nr.2.21. Funcţii şi criterii de utilizare a sistemelor de inteligenţă artificială în întreprinderi

Criterii / Funcţii	Există sarcini speciale?	Datele cerute sînt necesare?	Există procese complexe de decizie?	Se cer cunoştinţe de expert?
1) CONTABILITATEA FINANCIARĂ	D	D	D	D
2) CONTABILITATEA COSTURILOR	D	D	D	D
3) MARKETING - selectarea metodelor de previziune	D	D	D	D
4) PERSONAL - înlocuirea personalului indisponibil	D	D	D	D
5) PRODUCŢIE (PLANIFICARE ŞI CONTROL) (producţie cu calculator integrat: (CAM/CIM)				
a) programarea capacităţilor	D	D	D	D
b) ordonanţare	D	D	D	D
c) optimizare transporturi	D	D	D	D
d) selecţia furnizorilor	D	D	D	D

1) Wittemann, N., Parker, M.M, Gongla, P., (IBM Corporation), Enterprise-Wide Information Management Consultant: Use of Expert Systems in business administration functions, in "Economics and A.I.", Proceedings of the IFAC/IFORS/IFIP/IASC/AFCET Conference, Aix-en-Provence, France, 2-4 Sept. 1986, Vezi şi Hollingum, J., Expert Systems. Commercial Exploitation of AI, IFS Ltd., London, 1990 s.a.

Criterii / Funcții	Trebuie comparate strategii?	Este economic?	Este potrivită?
1) CONTABILITATEA FINANCIARĂ	D	D	D
2) CONTABILITATEA COSTURILOR	D	D	D
3) MARKETING - selectarea metodelor de previziune	D	D	D
4) PERSONAL - înlocuirea personalului indisponibil	D	D	D
5) PRODUCȚIE (PLANIFICARE ȘI CONTROL) (producție cu calculator integrat: (CAM/CIM)			
a) programarea capacităților	D	D	D
c) optimizare transporturi	D	D	D
d) selecția furnizorilor	D	D	D

2.6.4. Criteriile aplicării sistemelor expert

Pentru criteriile care stau la baza aplicării sistemelor expert există și alte opinii pe care le considerăm utile în acest context. Astfel, H.A.O. Krcmar de la City University of New York propune 13 criterii de judecare a justificării aplicabilității sistemelor expert în economie, organizate în trei grupe, după cum urmează:

1. Fezabilitatea și siguranța tehnică:
   - A. Structura solvabilă a problemei:
     - a) tipul problemei;
     - b) scopul problemei;
     - c) caracteristicile cunoașterii;
   - B. Mediul tehnic:
     - a) dezvoltarea și livrarea hardware-ului;
     - b) instrumentele software;
2. Disponibilizarea cunoașterii:
   - a) achiziționarea cunoașterii;
   - b) consensul experților implicați;
   - c) mentenabilitatea;
3. Situația agentului economic:
   - a) conștiința managementului;
   - b) acceptarea utilizatorilor;
   - c) costurile;
   - d) riscurile.

Primele două grupe de criterii (1 + 2) permit o evaluare a faptului dacă tehnologia sistemelor expert este capabilă să facă față mai bine decît tehnologia informatică obișnuită la o anumită problemă, dacă este mai nimerită și mai valoroasă într-un caz specific.

Ultima grupă (3) se referă la situația concretă a agentului economic.

Structura solvabilă a problemei are în vedere dacă problema necesită pricepere cognitivă, mai mult decît îndemînare procedurală (algoritmică). Sînt mai oportune pentru inteligența artificială problemele care presupun selecția de alternative pe baza unor criterii multiple. Intră în această categorie consultingul automat, diagnoza și planificarea, la care utilizatorul împreună cu sistemul de inteligență artificială lucrează pentru obținerea celor mai bune variante de soluții. În toate cazurile, se are în vedere o cît mai bună delimitare și definire a scopului problemei, în funcție de care se fixează și sarcinile concrete.

Caracteristicile cunoașterii depind de contextul problemei. De exemplu, dacă problema aparține managementului, atunci se vor include obligatoriu relațiile dintre mediu, componente și procese în afara definițiilor, separat pentru componente și separat pentru procese. Se vor avea în vedere componentele cu rol de monitorizare a mediului, întrucît acestea vor controla validitatea tuturor cunoștințelor și stabilitatea

cunoaşterii, în sensul că se va ţine seama de structura datelor, procedurile de management şi datele folosite curent. Procedurile de management sînt mai stabile şi se schimbă explicit iar datele constituie subiectul unor schimbări implicite şi rapide.

Pentru dezvoltarea şi livrarea sistemelor de inteligenţă artificială se utilizează întotdeauna un mediu tehnic adecvat. Problemele economice reclamă în mod obişnuit mari cantităţi de date şi echipamente care permit interogarea bazelor de date. La acestea se adaugă medii de programare adecvate. De exemplu, în prezent, se utilizează tot mai intens aşa-numitele Expert System Shell (limbaje vide), care trebuie să îndeplinească o serie de condiţii:

- prezentarea cunoaşterii să fie uşor de înţeles atît de către specialistul în cunoaştere cît şi de către utilizatori;
- utilizatorul să fie capabil să descopere ce ştie să facă sistemul;
- sistemul să fie capabil să se dezvolte dinamic odată cu disponibilizarea cunoaşterii de către expertul în domeniu;
- să aibă o interfaţă utilizator robustă şi flexibilă;
- să arate cum raţionează (să-şi explice raţionamentul).

Disponibilitatea cunoaşterii se referă la posibilităţile de obţinere şi procurare a cunoaşterii: experţii umani, literatura de specialitate, studii de caz etc. O condiţie importantă o constituie abilitatea experţilor de a adapta metodele şi abordările lor la soluţia problemei şi de a-şi exprima consensul asupra soluţiei. Nu întotdeauna consensul experţilor este necesar în mod real. De exemplu, pentru o problemă de management contabil trebuie intercorelate cunoştinţe foarte diverse, caz în care experţii vor identifica cel mai bun model necesar, arătînd şi cum trebuie asigurată mentenabilitatea cunoaşterii, mai ales prin componentele care monitorizează mediul şi provoacă schimbări în structura organizatorică, în programul de producţie etc.

Dezvoltarea unui sistem de inteligenţă artificială în domeniul gestiunii implică un mare consum de resurse şi o comunitate potenţială de utilizatori iar problema trebuie să fie suficient de critică pentru a obliga managementul să se decidă. Agenţii economici trebuie să dispună de climatul adecvat acestei tehnologii şi de utilizatori care o acceptă. La acestea să se adauge posibilitatea obţinerii beneficiilor altfel decît pe seama producţiei, prin folosirea potenţială a sistemului expert: îmbunătăţirea poziţiei în competiţie; reducerea riscurilor din cauza unor decizii mai bune; reducerea costurilor; cîştigarea unei noi priceperi, a unui control mai bun ş.a.

Costurile şi riscurile asociate unei aplicaţii concrete de inteligenţă artificială se pot micşora prin prototipizare şi prin metode de evaluare şi reducere cunoscute deja de la oricare alte implementări de tehnologii noi.

2.6.5. Aplicabilitatea inteligenţei artificiale în domeniul financiar-contabil

Din tabelul anterior 2.21 se desprind numai o parte a subfuncţiilor în care există deja implementări viabile: contabilitatea financiară (exterioară) şi contabilitatea managerială (analitică, a costurilor). În egală măsură putem menţiona: generarea rapoartelor de expertiză contabilă pe baza diagnozei făcută în prealabil; analiza bilanţului, analiza contului de profit şi pierderi; administrarea portofoliilor; planificarea impozitelor şi taxelor; evaluarea riscurilor; planificarea financiară; diagnoza financiară; evaluarea creditelor; prognozarea ratei de schimb valutar etc.

Tabelul nr. 2.22. Aplicaţiile în domeniul financiar-contabil

Denumirea şi anul implementării	DESTINAŢIA	Mediul de dezvoltare
TAXADVISOR (1982)	Planificarea impozitelor şi taxelor guvernamentale	EMYCIN
AUDITOR (1983)	Evaluarea fondurilor debitorilor greu solvabili	AL/X
ANSWERS (1984)	Decizii în timpul expertizei contabile	AL/X
PEGASE (1984)	Evaluarea fondurilor debitorilor greu solvabili	PROLOG
M1 (1984)	Diagnoza în vederea acordării creditelor	EMYCIN
AUDITPLANNER (-1984)	Decizii în procesul expertizei contabile	EMYCIN
MIME (1984)	Expertiza economico-financiară	SATIN

PANISSE (1984)	Prognoza ratei de schimb F.Fr - $ SUA	EMYCIN
COURTIER (1985)	Analiza informaţiilor de pe piaţa portofoliilor	EMYCIN
AIDE (1985)	Diagnoza financiară	APL
DACNOS (1985)	Realizarea funcţiilor contabile într-o reţea de calculatoare	LISP
EXFIN (1986)	Analiza şi selecţia planurilor financiare	Intelligence Service
FINEX (1986)	Diagnoza financiară	PROLOG
EwIMC (1986)	Planificare financiară. Analiza calităţii. Analiza valorii	IBM-ESE
CLA (1987)	Analiza creditelor comerciale	Syntelli-gence
CLU-E (1988)	Gestiunea poliţelor de asigurare	EXSYS Shell
SEAC (1988)	Diagnoza financiară	SNARK
Intellect (1988)	Planificarea financiară	AIC
Authorizer (1988)	Autorizarea creditelor	KEE LISP
LOAN RISKS (1988)	Evaluarea creditelor comerciale	AL/X
AY/ASQ (1989)	Decizii în procesul expertizei contabile	AL/X
XFC (1989)	Gestiunea informaţiei financiar-conta-bile	DEC VAX-/VMS
ISSUE (1990)	Gestiunea hîrtiilor de valoare	PROLOG

În domeniul celorlalte funcţii şi subfuncţii ale agentului economic remarcăm: planificarea producţiei; sistemele suport pentru decizii;

programarea producţiei; controlul fabricaţiei; generarea standardelor de calitate şi controlul calităţii; planificarea investiţiilor; administrarea orelor flexibile de lucru; planificarea ajutoarelor; selectarea serviciilor de diverse tipuri; biblioteconomie; administrarea banilor; planificarea pensionărilor; gestiunea poliţelor de asigurare; prognoza micro- şi macroeconomică; analiza competitivităţii firmei ş.a.

În prezent, cele mai răspîndite sisteme de inteligenţă artificială sînt în domeniul serviciilor financiar-bancare şi de asigurări (vezi tabelul 2.22).

Eventualele sisteme de inteligenţă artificială descentralizate depind numai de filosofia generală a organizării sistemului informatic. Se prevede existenţa unor sisteme de inteligenţă artificială implementate pe microcalculatoare compatibile IBM PC, XT, AT, PS-2, dotate cu facilităţi de acces la baza de date a sistemului informatic al întreprinderii. Se urmăreşte crearea aceleiaşi interfeţe utilizator atît pentru funcţiile rezolvate cu programe inteligente cît şi pentru cele rezolvate cu programe algoritmice. Sistemul de inteligenţă artificială este văzut ca o parte componentă a sistemului informatic al agentului economic.

Ca tendinţe, trebuie menţionate proiectarea şi implementarea de sisteme multiexpert, începută deja în marile companii; proiectarea de sisteme hibride (care includ atît produse program algoritmice cît şi programe inteligente) şi funcţionarea lor în concepţie integrată; reîntoarcerea la limbajul COBOL şi mediile comune pentru aplicaţiile de inteligenţă artificială.

În concluzie, în regiile autonome, societăţile comerciale şi ceilalţi agenţi economici din ţara noastră, proiectarea sistemelor informatice a devenit dependentă de proiectarea sistemelor bazate pe cunoştinţe (Knowledge Based Systems), unde vor coexista sisteme expert, fie separat, fie integrate în sistemele informatice. Toate asemenea implementări fac obiectul unor cercetări asidui în scopul obţinerii de soluţii eficiente, costuri reduse şi fără riscuri. În prezent tehnologiile de inteligenţă artificială fac obiectul unor finisări intense, personalul este reciclat în asemenea tehnologii şi se acţionează încă în sensul atitudinii tradiţionale. Ele vor avea un impact deosebit asupra funcţiilor executive şi-şi vor pune amprenta pe toate deciziile.

2.6.6. Atributele sistemelor de inteligenţă artificială

Atributele sistemelor de inteligenţă artificială în cazul agenţilor economici se pot prezenta

astfel:

- abilitatea de a asista experţii la proiectarea propriilor sisteme expert şi utilizatorii la desfăşurarea activităţilor specifice;

- sensibilizarea managementului asupra schimbărilor operaţiilor şi relaţiilor dintre funcţii;

- autoorganizarea şi restructurareă bazelor de cunoştinţe şi relaţiilor, prin activarea automată în condiţiile unor factori critici şi generarea de semnale asupra operaţiilor generatoare de schimbări în direcţia reechilibrării;

- integrarea dintre sistemul de colectare a datelor, cu modulele operaţionale şi procedurile cu specific inteligent pentru analiza economico-financiară.

Managementul executiv în era sistemelor inteligente va consuma mai mult timp cu strategia şi modalităţile de integrare a operaţiilor, şi mai puţin timp cu controlul operaţiilor, care va fi asigurat de către sistemele de inteligenţă artificială integrate. Tot mai mulţi specialişti din economie şi oameni de afaceri văd în această tehnologie o nouă sursă de creştere a productivităţii muncii, pentru faptul că oferă siguranţă şi procură multă experienţă şi inteligenţă prin combinarea contabilităţii manageriale, analizelor statistice, analizelor economico-financiare, cu luarea deciziilor în condiţiile unor structuri şi strategii prestabilite şi în dinamică.

Extrema specializare a aplicaţiilor sistemelor de inteligenţă artificială tinde să ne demonstreze generalizarea acestei tehnologii la toate domeniile dar, tentativa de a le aplica la domenii mai largi de activitate obligă proiectanţii să elaboreze sisteme multiexpert[1]. Asemenea sisteme sînt concepute fie pentru rezolvarea problemelor multiple dintr-un singur domeniu, fie pentru rezolvarea problemelor multidisciplinare. În prezent, teoria sistemelor multiexpert a deschis o nouă direcţie a preocupărilor ştiinţifice.

[1]Ericson, C.E., ş.a., Expert Systems Application in Integrated Network Management, Artech House, 1989 ş.a.

CAPITOLUL III

TEORIA ŞI DEZVOLTAREA SISTEMELOR EXPERT ÎN CONTABILITATE

Introducere De la mijlocul anilor 70 un anumit număr de produse-program de inteligenţă artificială sînt considerate sisteme expert. Anul 1964 este consacrat drept an de început al sistemelor expert, datorită elaborării programului DENDRAL pentru enumerarea şi notarea moleculelor organice, ca structuri arborescente, la Stanford University din SUA.

Am văzut anterior că sistemele expert constituie un domeniu major de aplicaţii, alături de recunoaşterea formelor şi prelucrarea limbajului natural.

Sistemele expert suscită interes în cele mai diverse medii ale specialiştilor, tocmai pentru că permit informatizarea anumitor funcţii intelectuale calificate, toate deosebit de utile în activitatea agenţilor economici şi dificile de modelat sub forma produselor-program algoritmice, de genul:
- identificarea şi diagnosticarea situaţiilor;
- previziunea evenimentelor;
- conceperea obiectelor de orice tip;
- planificarea acţiunilor;
- decizii în condiţii de risc şi incertitudine etc.

În cele mai diverse situaţii, dificultatea soluţionării unor asemenea probleme complexe poate fi înlăturată prin metodologia sistemelor expert.

Caracteristica esenţială a metodologiei sistemelor expert în raport cu alte tehnologii de inteligenţă artificială, constă în încercarea de a reproduce facultăţile de a decide sau de a judeca ale experţilor umani, în

sensul că experţii umani pot emite cunoştinţe perfect structurate specifice unui domeniu de aplicaţii. În prezent, metodologia sistemelor expert a depăşit stadiul incipienţei. Pentru domeniul contabilităţii, metodologia sistemelor expert oferă numeroase promisiuni. Ea înseamnă dezvoltarea previzibilă a tehnologiei informatice în direcţia realizării de sisteme care integrează produse-program inteligente şi produse-program algoritmice, întrucît o parte însemnată din prelucrările specifice contabilităţii vor rămîne prelucrări de date.

Prezentăm în continuare, printre primii în literatura de specialitate românească, teoria şi dezvoltarea sistemelor expert în contabilitate.

3.1. Noţiunea de sistem expert

În mai puţin de un deceniu, sistemele expert au evoluat de la poziţia relativ obscură la aceea de instrumente practice importante pentru întreprinderi. Sistemele expert sînt astăzi o tehnologie care permite societăţilor comerciale şi regiilor autonome să prelucreze cunoştinţe cu ajutorul calculatoarelor. Directorii adoptă această tehnologie din cauză că înţeleg că, astăzi, întreprindere modernă şi rentabilă este aceea care obţine cea mai bună eficienţă prin manipularea şi stăpînirea cunoaşterii. Primul lucru care trebuie spus despre sistemele expert este că ele sînt construite printr-o tehnologie bine stabilită. Al doilea lucru important constă în faptul că sistemul expert constituie o adăugare importantă la calculatoarele existente a unei mulţimi de tehnici complementare faţă de cele tradiţionale, care sînt foarte puternice: noi limbaje de programare, noi medii de programare, noi echipamente şi software specializate pentru implementarea sistemelor expert. Al treilea lucru se referă la faptul că sistemele expert cer o nouă implicare din partea utilizatorilor, fie că-şi dezvoltă propriile sisteme expert, fie că le folosesc pe cele comercializabile. Ca urmare, nimeni nu va putea sta indiferent la această tehnologie de vîrf. Acum este cazul să vedem cum poate fi abordată noţiunea de sistem expert.

Edward Feigenbaum[1] de la Stanford University arată că sistemele expert sînt programe capabile să raţioneze la nivelul unui expert uman. Din această definiţie rezultă trei idei succesive:

- sistemele expert dispun de cunoştinţe şi de capacitatea de a desfăşura activităţi intelectuale umane;
- sistemele expert sînt organizate pentru achiziţia şi exploatarea cunoaşterii dintr-un domeniu particular;
- sistemele expert dispun de metode de invocare a cunoaşterii pentru exprimarea expertizei, comportîndu-se ca un "asistent inteligent".

1 Feigenbaum, E., McCorduck, P., The Fifth Generation, Addison-Wesley Publishing Co., 1983, p. 18.

Şi alte definiţii sînt importante pentru mai buna conturare a noţiunii de sistem expert. Prezentăm în ordine cîteva asemenea definiţii. Buchanan, B. şi Davis, R. ş.a. de la Stanford University arată că sistemele expert sînt:

"...programe informatice care încorporează cunoaşterea specializată şi experienţa unui expert uman; acestea sînt rezultatul ingineriei cunoaşterii. Ingineria cunoaşterii se interesează de realizarea programelor capabile de performanţe la nivelul experţilor, deoarece ele acumulează mari cantităţi de cunoştinţe relative la un domeniu particular de aplicaţii...".

Agapeyeff, A. de la British Computer Society Expert Systems arată că "...un sistem expert rezultă din implementarea pe calculator a unei baze de cunoştinţe astfel încît maşina să poată da avize inteligente sau să ia decizii inteligente... O caracteristică suplimentară, de aşteptat, fundamentală, constă în atitudinea sistemului de a justifica la cerere propria linie de raţionament, într-o manieră direct inteligibilă. Stilul adoptat pentru atingerea acestor caracteristici este programarea pe baza regulilor de producţie."

"Sistemul expert este un program care posedă o mare cantitate de cunoştinţe într-un domeniu specializat, provenite de la un expert uman, fiind capabil să atingă performanţele expertului în domeniu."[1]

"Sistemele expert sînt programe, dar pot fi tot atît de bine maşini cu software, destinate să înlocuiască sau să asiste specialistul în domeniile unde este recunoscută necesitatea expertizei umane."[2]

Această din urmă definiţie este considerată de autorul ei drept funcţională, ea aduce unele elemente de noutate fiind mai apropiată de rolul şi obiectivele sistemelor expert. Oricum, cercetătorii sînt de acord cu faptul că, sub denumirea de sistem expert se află acele programe de

[1]Bonnet, A., ş.a., Systems Experts: vers la maitrise tèchniques InterEditions, Paris, 1986, p.14

[2]Farreny, H., Les Systèmes Experts. Principes et exemples, Cepadues Editions, Toulouse, 1986, p. 17.

inteligenţă artificială, bazate pe o cunoaştere de înalt nivel, comparabilă cu a celor mai competenţi specialişti dintr-un domeniu aplicativ, în care aceste programe pot realiza performanţe de gîndire şi intuiţie, similare experţilor umani.

Există în literatura de specialitate şi opinii potrivit cărora "denumirea de sistem expert este inexactă şi greşită, în plus, ea aduce un nefericit amestec de mister şi magie. Mai nimerită este denumirea de Knowledge Based Systems, care descrie mai fidel domeniul şi tehnicile de programare puternice care s-au descoperit încă din perioada pionieratului sistemelor expert. Este mai puţin emotivă şi mult mai exactă. Knowledge Based Systems nu folosesc acelaşi proces de raţionament ca experţii umani, ele nu gîndesc şi nici nu imită expertiza umană. Ele doar memorează cunoaşterea, care este pusă în ele de către oameni...şi deci, pe aceasta o manipulează, o foloseşte pentru a infera rezultatul...".[1]

Sîntem de acord cu faptul că centrul funcţional al sistemelor expert îl reprezintă colecţia de cunoştinţe, structurate, organizate şi reprezentate în mod adecvat. În scopul relevării relaţiei cu sistemele de inteligenţă artificială, unii autori arată că sistemele expert sînt sisteme bazate pe cunoştinţe (Knowledge Based Systems), cum le prezentăm în fig. 3.1., deoarece şi ele prelucrează cunoaşterea organizată în baze de cunoştinţe.[2]

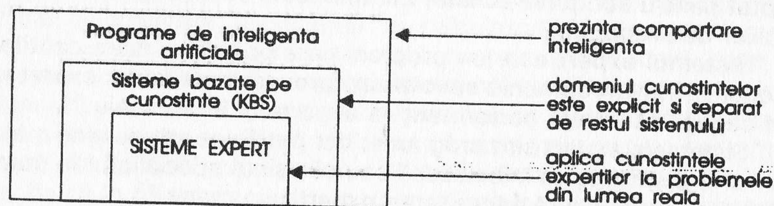


Figura 3.1. Relaţia dintre programele de inteligenţă artificială, sistemele bazate pe cunoştinţe şi sisteme expert

[1]Hollingum, J., Op. cit., p.7

[2]Waterman, A.D., A Guide to Expert Systems, Addison - Wesley Pub. Co. Reading, Massachusetts, 1986.

3.2. Obiectivele, structura de bază şi caracteristicile sistemelor expert

Sistemele expert se dezvoltă cu ajutorul unei metodologii informatice urmărind trei obiective comune:

1. Achiziţionarea uşoară a pieselor de cunoaştere, prin exprimarea cît mai direct posibilă a regulilor obţinute de la experţi;

2. Exploatarea eficientă a colecţiei pieselor de cunoaştere (fapte şi reguli) prin:

2.1. combinarea şi/înlănţuirea regulilor pentru a infera cunoştinţe prin judecăţi, planuri, demonstraţii, decizii, predicţii, noi reguli etc.;

2.2. luarea în seamă a modului în care noile cunoştiinţe sînt inferate;

3. Să suporte cu uşurinţă ansamblul operaţiilor asupra pieselor de cunoaştere şi să permită adăugarea, modificarea şi eliminarea faptelor şi regulilor.

În concordanţă cu aceste obiective, **componentele de bază ale unui sistem expert** sînt:

a) O bază de cunoştinţe, pentru stocarea pieselor de cunoaştere specifice unui domeniu aplicativ, creată şi organizată pentru satisfacerea obiectivului nr.3;

b) Motorul de inferenţe (maşina deductivă), o secvenţă de piese de cunoaştere operatorie, care exploatează baza de cunoştinţe şi este destinată satisfacerii obiectivului nr.2.1.;

c) Interfaţa de dialog cu utilizatorii, care dispune de un limbaj de exprimare a cunoaşterii achiziţionate de la experţii umani.

Pentru achiziţia şi modificarea pieselor de cunoaştere (obiectivele 1 şi 3), pentru colectarea informaţiei despre problemă, asigurarea unei interacţiuni eficiente cu utilizatorul în timpul lucrului, ca şi pentru luarea în seamă a mecanismului de raţionament (obiectiv 2.2.), un sistem expert trebuie să asigure de asemenea funcţii complementare de dialog şi explicare a propriului comportament. În fig. 3.2. prezentăm structura de bază a unui sistem expert.

Fără îndoială, originalitatea metodologiei sistemelor expert constă în existenţa celor trei componente şi a relaţiilor dintre ele: baza de cunoştinţe, motorul de inferenţe şi interfaţa utilizator.

Există două categorii de utilizatori:

- utilizatori experţi, cei care introduc reguli în baza de cunoştinţe sau care crează şi actualizează această bază;

- utilizatori comuni, cei care poartă un dialog cu sistemul expert ca

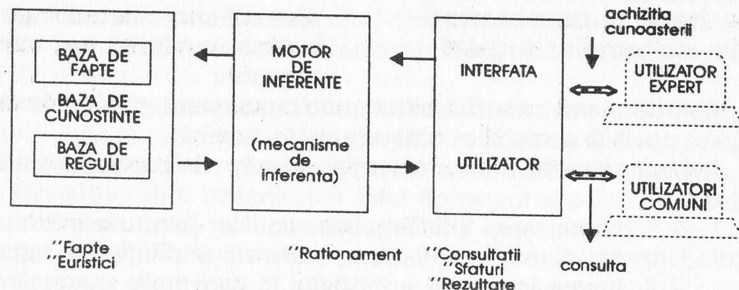


Figura 3.2. Structura de bază a unui sistem expert

şi cum ar consulta un expert uman.

Baza de cunoştinţe constă din fapte şi euristici asociate problemei, care descriu situaţii evidente, reale sau ipotetice, precum şi reguli sau piese de cunoaştere operatorie. Regulile sînt exprimate cu ajutorul unor expresii adecvate înţelegerii utilizatorilor. Anumite sisteme expert folosesc şi o bază de date relaţională, în care relaţiile dintre variatele obiecte şi evenimente sînt memorate explicit, pentru o mai bună flexibilitate a memorării şi regăsirii. În acest caz, trebuie să existe şi o interfaţă pentru date.

Motorul de inferenţe este un program (poate fi chiar şi un circuit integrat microprogramat) care pune în lucru mecanisme generale inferenţiale, de combinare a faptelor şi regulilor din baza de cunoştinţe. Conform variatelor strategii de control, în strînsă legătură cu domeniul aplicativ, motorul de inferenţe selectează reguli, le interpretează şi le înlănţuieşte logic pentru satisfacerea condiţiilor. În principiu, mecanismul inferenţial determină modificări în baza de cunoştinţe, atît în baza de fapte cît şi în baza de reguli. Rezolvarea problemei, în final, se va concretiza sub forma unor propoziţii, a unui diagnostic sau plan de acţiuni. Motorul de inferenţe are de fapt două **componente principale**: sistemul de administrare a bazei de cunoştinţe şi procesorul (de inferenţe) simbolic.

Sistemul de administrare a bazei de cunoştinţe administrează, efectuează operaţii de organizare automată, control şi actualizare a pieselor de cunoaştere, şi iniţiază căutări pentru controlul relevanţei pe

linia de raţionament pe care lucrează procesorul de inferenţe simbolic. La rîndul său, procesorul de inferenţe simbolic oferă o metodă de prelucrare, prin care se furnizează liniile de raţionament. Atunci cînd cunoştinţele şi datele din lumea reală nu sînt exacte, anumite metode de inferenţă pot utiliza grade de incertitudine în derularea mecanismului inferenţial.

Interfaţa utilizator este o altă componentă critică a sistemului expert. Prin intermediul său este posibil accesul utilizatorilor la faptele, datele şi regulile din baza de cunoştinţe, se permite achiziţia cunoaşterii de la experţi, precum şi dialogul cu ceilalţi utilizatori ai sistemului, iar uneori chiar cu alte sisteme. Din acest motiv, interfaţa trebuie să fie cît mai naturală şi prietenoasă, folosind un limbaj cît mai apropiat de limbajul natural, cu texte şi imagini afişate la o viteză confortabilă pentru utilizatori.

Dialogul cu sistemul expert trebuie să asigure trei moduri utilizator:

-pentru utilizatorii beneficiari, clienţi, care cer şi obţin sfaturi sau consultaţii şi răspunsuri la problemele puse;

-pentru utilizatorii experţi, instructori, care îmbogăţesc cunoaşterea, introducînd noi fapte şi reguli în baza de cunoştinţe;

-pentru utilizatori studenţi/şcolari/personal care se instruiesc, prin aflarea unor cunoştinţe noi, sau cărora li se evaluează în final nivelul de cunoaştere.

Evident, este de dorit o interfaţă în limbaj natural pentru folosirea sistemului expert în toate cele trei moduri utilizator.

În sistemele expert mai complexe, se include un modul de explicare (explicativ) în scopul de a permite utilizatorului să obţină explicaţii asupra proceselor inferenţiale, soluţiilor obţinute, ori pentru evidenţierea unor piese de cunoaştere eronate sau care lipsesc, evidenţierea cauzelor unor eşecuri etc.

Trebuie reţinut că sistemele pot acţiona ca depozite sinergetice de cunoştinţe achiziţionate de la o varietate de experţi într-un domeniu particular. Combinarea cunoaşterii în asemenea situaţii poate permite utilizatorilor un nivel de expertiză care depăşeşte capacitatea unui singur expert.

Atunci cînd cunoaşterea este memorată sub forma regulilor de producţie, baza de cunoştinţe este numită **bază de reguli** iar motorul de inferenţe se numeşte **interpretor de reguli**.

Sistemele expert se dezvoltă prin analiza îngrijită a cunoaşterii experţilor în domeniu, de către o persoană specializată în inteligenţa artificială numită "inginer de cunoaştere" sau "cognotician". Această persoană,

experimentată în proiectarea şi construirea sistemelor expert, acţionează în conformitate cu metodologia de dezvoltare specifică şi obţine sisteme expert capabile de performanţe identice cu ale expertului uman. Dar acest lucru nu este suficient. De aceea sistemul expert trebuie să se caracterizeze prin oferta de soluţii optime într-o perioadă de timp foarte redusă.

De altfel, pe lîngă această caracteristică, sistemele expert trebuie să îndeplinească şi alte caracteristici (fig.3.3.).

Deci, principala caracteristică a sistemelor expert este priceperea sau îndemînarea de a obţine soluţii eficiente folosind cele mai scurte căi de raţionament, cu eliminarea rapidă a regulilor incorecte. Un sistem expert manipulează de fapt diferitele simboluri specifice pieselor de cunoaştere, reprezentate în baza de cunoaştinţe, dar sînt capabile să efectueze şi calcule complexe, de aceea el trebuie să fie robust, adică să nu aibă îndemînare numai într-un subiect anume, ci, să folosească cunoştinţe generale şi metode de rezolvare a problemelor, pentru a infera după principii proprii cînd i se oferă fapte, date, şi reguli incomplete.

SISTEME EXPERT

- **Expertiza**
  - **performanţă de expert;**
  - **nivel ridicat de calificare;**
  - **robusteţe adecvată;**
- **Raţionament simbolic**
  - **reprezentarea simbolică a cunoaşterii;**
  - **reformularea cunoaşterii simbolice;**
- **Profunzime**
  - **lucrează cu probleme dificile;**
  - **utilizează reguli complexe;**
- **Autocunoaştere**
  - **examinează propriile raţionamente;**
  - **explică operaţiile şi acţiunile;**

Figura 3.3. Caracteristicile sistemelor expert.

Sistemele expert se caracterizează prin profunzime, în sensul că pot opera într-un domeniu relativ îngust, în care soluţionează probleme dificile, fără să posede calitatea de reformulare a problemelor. Regulile sînt în mod necesar complexe iar numărul lor este ridicat. Sistemele

expert posedă o cunoaştere care le permite să infereze asupra propriilor operaţii şi să verifice calitatea şi consistenţa concluziilor/soluţiilor oferite. Cunoştinţele de care dispun sistemele expert, cu privire la desfăşurarea mecanismului inferenţial, poartă denumirea de "metacunoştinţe".

Sistemele expert sînt dotate cu posibilităţi de explicare a operaţiilor, a modului cum au ajuns la soluţiile proprii. Explicaţiile sînt oferite prin afişarea lanţurilor de inferenţe, ca urmare a abilităţii de examinare a fiecărei reguli din lanţul inferenţial. Explicarea este un aspect important a autocunoaşterii dar nu şi suficient. Autocunoaşterea prezintă importanţă datorită faptului că utilizatorii capătă mai multă încredere, sistemul poate fi mai uşor depanat şi oferă uşurinţă în prevederea efectului schimbărilor.

3.3.Motoare de inferenţe

3.3.1. Ciclul de bază al unui motor de inferenţe

Am văzut în capitolul anterior că mecanismul inferenţial se desfăşoară sub forma ciclurilor de bază. Este vorba de etapele de EVALUARE şi EXECUŢIE a regulilor.

Etapa de EVALUARE cuprinde trei faze: SELECŢIA, FILTRAREA şi REZOLVAREA CONFLICTELOR.

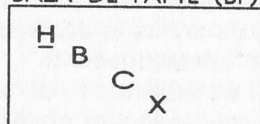
Un motor de inferenţe, odată lansat, în etapa de EVALUARE, determină dacă există în baza de reguli curentă regulile de declanşat şi dacă acelea sînt corespunzătoare, dacă au partea de condiţie satisfăcută. În etapa de EXECUŢIE, motorul declanşează regulile reţinute în timpul evaluării, acţionînd asupra bazei de cunoştinţe. În continuare, nu vom mai relua prezentarea integrală a ciclului de bază, însă credem că prezintă interes unele principii de funcţionare a motoarelor de inferenţă, pe care trebuie să le abordăm pe un exemplu concret, al unui sistem expert de dimensiuni foarte reduse, numit MICROSIMBOL.[1]

MICROSIMBOL dispune de un motor de inferenţe numit MICROMOTOR, care funcţionează pe o bază de cunoştinţe reprezentată simbolic şi în care se observă faptele evidente, stabilite B C X, faptul de demonstrat sau de stabilit <u>H</u>, împreună cu regulile sau piesele de cunoaştere operatorie, grupate în baza de reguli (fig.3.4.).

Să trecem acum în revistă, pe exemplul lui MICROMOTOR, etapele

---

[1] **Adaptare după Farreny, H., Op. cit., p.40 ş.u.**



BAZA DE REGULI (BR)

1	P	→	B, D, E
2	A, B	→	D, G
3	A	→	C, P, K
4	D, L	→	C, Y
5	E	→	M, D
6	H	→	L, A
7	H, W	→	Z
8	P, L	→	W

Figura 3.4. Baza de cunoştinţe iniţială pentru MICROMOTOR

ciclului de bază a motoarelor de inferenţe (vezi şi schema ciclului de bază din fig. 2.18.).

a) SELECŢIA (RESTRICŢIA)

În timpul primului ciclu, nu se operează nici-o selecţie, şi atunci, subansamblul R1 este identic cu BR (baza de reguli) iniţială, iar subansamblul F1 este similar cu BF (baza de fapte) iniţială. În principiu, dacă la ciclul de rang n, o regulă s-a executat (adică R3 este non vid) nici o selecţie (restricţie) nu va fi operată la ciclul de rang n+1, ceea ce va avea ca rezultat R1 ≡ BR şi F1 ≡ BF din ciclul n. Dimpotrivă, dacă la ciclul n, subansamblul R3 a fost vid, atunci la ciclul n+1 se vor afecta valori din BR şi BF iniţiale, ale ultimului ciclu anterior, care a determinat execuţia unei reguli (ciclul de rang.n - 1 sau inferior). Se spune că are loc o întoarcere-înapoi (backtracking) la contextul ciclului n - 1. Dacă nu există ciclul anterior (n-1), motorul se opreşte.

b) FILTRAREA

Este ştiut că în membrul stîng al regulii se află condiţiile de declanşare, iar acestea sînt satisfăcute numai atunci cînd fiecare din simbolurile prezentate în membrul stîng fac parte şi din baza de fapte. În situaţia concretă din exemplul nostru, iniţial numai regula 6 poate fi aplicabilă (vezi BR iniţială din fig. 3.4.). În realitate, satisfacerea condiţiilor de declanşare este definită mai subtil. De pildă, se poate folosi o expresie de forma "CONT-200 este cont-de-activ" în loc de H din baza de fapte, şi atunci e mai natural de afirmat că declanşatorul "CONT-200 este cont-de-activ" este satisfăcut prin faptul "CONT-200 este cont-de-stocuri".

c) REZOLVAREA CONFLICTELOR

Să ne reamintim că subansamblul de reguli aplicabile după faza de filtrare constituie "ansamblul de conflict" notat cu R2, iar rezolvarea conflictelor produce pe R3, un subansamblu al lui R2, care reuneşte

regulile de declanşat efectiv. În MICROMOTOR convenim ca, sistematic, numai prima regulă din R2 să fie plasată în R3; ordinea aşezării regulilor în R2 are la bază ordinea de realizare a filtrării, adică ordinea fixată de la 1 la 6 a regulilor din BR.

O asemenea strategie de rezolvare a conflictelor este frecvent utilizată în sistemele expert reale, capabile de "întoarcere-înapoi" (backtracking). Trebuie să menţionăm în acest punct al expunerii că, motoarele de inferenţe pot funcţiona în două **regimuri de control:**

- regimul de control irevocabil;
- regimul de control prin tentative.

Regimul de control irevocabil (irrevocable control regime) este atunci cînd motorul de inferenţe reacţionează identic în două situaţii:

- Cînd R3 este vid şi motorul se opreşte;
- Cînd R3 este vid şi se reconsideră ansamblul de conflicte R2 dintr-un ciclu anterior (n - 1), reexaminîndu-se posibilitatea declanşării altor reguli din R2. Dacă o declanşare de reguli nu este posibilă nici în această situaţie, atunci motorul de inferenţe se opreşte.

Regimul de control prin tentative (tentative control regime) este atunci cînd există posibilitatea înlocuirii declanşatorilor din reguli cu alţii, şi se poate relua rezolvarea conflictelor anterioare, reîncercîndu-se declanşarea regulilor, continuînd în maniera aceasta funcţionarea motorului. Realizarea "întoarcerii-înapoi" se poate prezenta astfel: în timpul ciclului de rang n se formează subansamblul R3 vid şi nu are loc etapa de EXECUŢIE; la ciclul de rang n + 1 faza de SELECŢIE impune direct subansamblul R1 a ciclului n - 1 şi faza de FILTRARE produce ansamblul R2 al ciclului n - 1. În acest moment, regula care era reţinută prin REZOLVAREA CONFLICTELOR, tot din ciclul n - 1, va fi automat eliminată din R2 a ciclului n + 1. Cu alte cuvinte, dacă în ciclul n s-a ajuns la un impas, are loc o întoarcere la ciclul n + 1, dar în contextul ciclului n - 1.

MICROMOTOR din exemplul nostru, funcţionează numai în regim de control prin tentative, ceea ce înseamnă că, dacă alegerea sistematică a primei reguli selectate conduce la un ciclu de bază de rang n, pentru care ansamblul de conflict este vid, faza de SELECŢIE a ciclului n + 1 va restaura starea BF care exista la începutul ciclului n - 1. Faza de FILTRARE a ciclului n+1 este evitată prin recuperarea ansamblului de conflict R2, determinat de ciclul n - 1, iar faza de REZOLVARE A CONFLICTELOR din ciclul n + 1 reţine a doua regulă din R2, dacă există, în caz contrar are loc întoarcerea la ciclul anterior, n - 2.

Adoptarea acestei strategii de control în MICROMOTOR oferă de fapt

două variante:

1) În filtrarea efectuată pe R2, prima regulă din R2 să se treacă în etapa de execuţie, fiind scoasă din R2, şi în caz de întoarcere la acest ciclu este suficientă selectarea unei noi prime reguli din R2, dacă există;

2) Filtrarea este limitată la căutarea primei reguli aplicabile, se execută întoarcerea la ciclul în care a doua regulă este cercetată.

MICROMOTOR urmează prima variantă.

EXECUŢIA constă în introducerea în BF a simbolurilor care figurează în membrul drept a regulii şi eliminarea din BF a simbolurilor care figurează în membrul stîng, reprezentative pentru faptul de stabilit. De exemplu, execuţia regulii 6, considerînd BF iniţială, va produce o nouă BF care reuneşte faptele B, C, X, L, şi A. În acest caz, eliminarea lui H, pentru introducerea lui A este o strategie particulară de raţionament, care ia seamă de faptul că obţinerea unei justificări pentru ipoteza H poate fi înlocuită cu căutarea unei justificări pentru ipoteza A.

Trebuie remarcate unele completări asupra EXECUŢIEI, şi anume:

- dacă membrul stîng al regulii de declanşat comportă un simbol F sau F, atunci cînd F este deja prezent în BF, nu se va mai adăuga nici un alt F sau F în BF;
- dacă membrul drept al regulii conţine pe F, atunci, cînd F este deja prezent în BF, nu se va mai adăuga un alt F;
- dacă membrul drept al regulii conţine pe F, atunci cînd F este deja prezent în BF, se adaugă F şi se elimină F.

Condiţiile de oprire ale MICROMOTOR-ului sînt următoarele:

- cînd BF curentă nu conţine simboluri subliniate, adică fapte de stabilit sau ipoteze de justificat, se consideră că toate faptele au fost stabilite (demonstrate), şi se produce o oprire interpretată ca un "succes";
- dacă prin întoarcerea-înapoi se constată că, în final, BF curentă mai conţine încă cel puţin un fapt de stabilit, are loc o oprire a motorului, interpretată ca un "eşec".

Ţinînd seama de aceste precizări, să urmărim în continuare o **sesiune completă cu MICROSIMBOL**, care se desfăşoară conform grafului din fig. 3.5., cu un lanţ inferenţial de zece cicluri de bază.

Primul ciclu

În primul ciclu numai regula 6 este declanşabilă, deci se va obţine R2 = {6} şi în urma execuţiei sale BF = {L, A, B, C, X}. Se elimină H şi se introduce L şi A. După execuţia acestei reguli, ansamblul de conflict R2 va rămîne vid. Să ne imaginăm o zonă de memorie de lucru care se

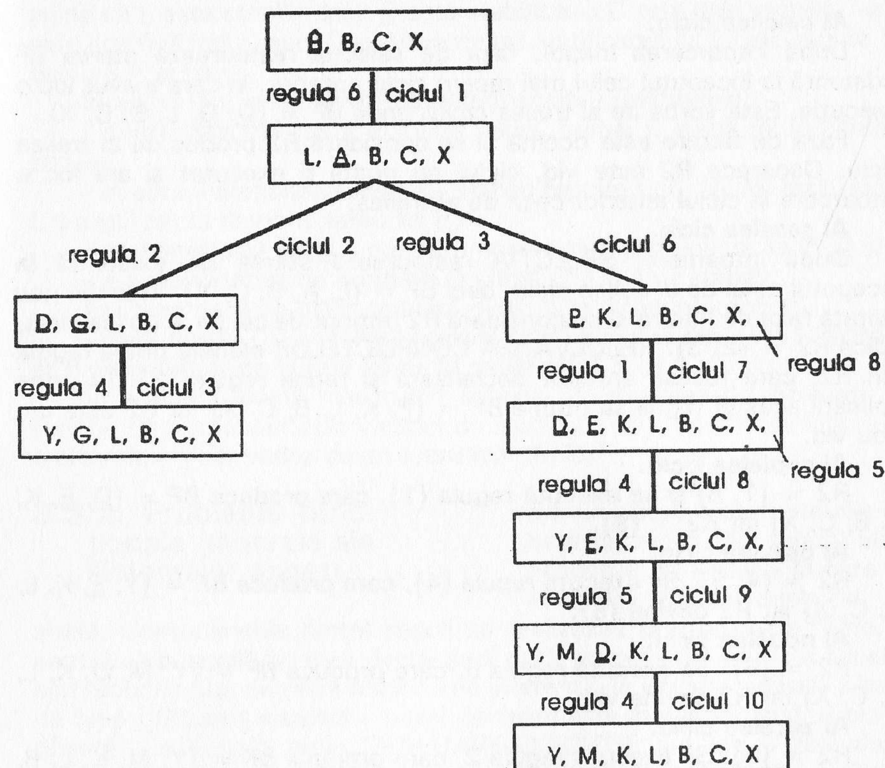


Figura 3.5. Înlănţuirea ciclurilor inferenţiale realizată de către MICROMOTOR. Ciclurile 4 şi 5 nu s-au reprezentat

reiniţializează cu zero, pentru acest ansamblu.

Al doilea ciclu.

Noul ansamblu R2 obţinut indică ordinea de filtrare {2, 3} şi se va da prioritate regulii 2. Rezultă în final BF = {D, G, L, B, C, X}. S-a eliminat A deoarece se află în stînga regulii 2 iar în R2 se memorează {3} întrucît regula 2 tocmai s-a executat. Deci R2 = {3}.

Al treilea ciclu.

R2 = {4}, de unde BF = {Y, G, L, B, C, X} iar R2 devine vid.

Al patrulea ciclu.

R2 este vid. Ciclul nu poate intra în execuţie şi are loc o întoarcere la ciclul următor.

Al cincilea ciclu.

După întoarcerea înapoi, faza de selecţie restaurează starea BF existentă la începutul celui mai recent ciclu anterior, în care a avut loc o execuţie. Este vorba de al treilea ciclu, unde BF = {D, G, L, B, C, X}.

Faza de filtrare este ocolită şi se consideră R2 produs de al treilea ciclu. Deoarece R2 este vid, ciclul nu poate fi executat şi are loc o întoarcere la ciclul anterior celui de al treilea.

Al şaselea ciclu.

După întoarcere, SELECŢIA restaurează starea BF existentă la începutul celui de al doilea ciclu, deci BF = {L, A, B, C, X}. Este din nou evitată faza de filtrare şi se consideră R2 obţinut de cel de al doilea ciclu, adică R2 = {2, 3}. REZOLVAREA CONFLICTELOR elimină prima regulă din R2, care fusese anterior declanşată şi reţine regula {3}. În urma aplicării acestei reguli se obţine BF = {P, K, L, B, C, X} iar R2 este din nou vid.

Al şaptelea ciclu.

R2 = {1, 8} şi se execută regula {1}, care produce BF = {D, E, K, L, B, C, X} iar R2 = {8}.

Al optulea ciclu.

R2 = {4, 5}. Se execută regula {4}, care produce BF = {Y, E, K, L, B, C, X} iar R2 devine {5}.

Al nouălea ciclu.

R2 = {5}. Se execută regula 5, care produce BF = {Y, M, D, K, L, B, C, X} iar R2 devine vid.

Al zecelea ciclu.

R2 = {4}. Se execută regula 2, care produce BF = {Y, M, K, L, B, C, X}, iar R2 rămîne vid. În acest moment MICROMOTOR constată că nu mai există nici un fapt de stabilit (ipoteza de dovedit) şi se opreşte. Aplicarea regulilor 6, 3, 1, 4, 5 şi 4 a permis stabilirea faptului H (ipoteza în baza de fapte iniţială) în funcţie de faptele deja stabilite B, C, X.

Înlănţuirea acestor reguli în EXECUŢIE (ramura din dreapta schemei din fig. 3.5.) se interpretează astfel:

- H este demonstrat dacă există A;
- A este demonstrat dacă există P;
- P este demonstrat dacă există D şi E;
- D este demonstrat dacă L este stabilit, dacă C este stabilit sau C este deja stabilit, fapt care dovedeşte pe C, deci şi pe D (urmăriţi regula 4);
- E este demonstrat dacă există D şi, în plus, D este demonstrat ştiind că L este stabilit dacă C este stabilit sau C este deja stabilit, lucru care justifică întîi C apoi D şi influenţează justificarea lui P (vezi regula 1), deci pe A şi pe H.

În acest lanţ inferenţial trebuie observate următoarele:

- dintre cele trei fapte stabilite iniţial, numai C joacă rol în demonstraţie;
- în cursul demonstraţiei s-au stabilit faptele L, K, Y, M, însă numai L s-a utilizat la demonstrarea lui H;
- demonstrarea lui H a permis trecerea prin justificarea lui P, D şi E;
- s-a demonstrat de două ori D ştiind că L a fost stabilit.

Era bine dacă se evita această dublare în ciclul inferenţial;

- dacă regula {8} ar fi fost aşezată în BR înaintea regulii {1}, motorul ar fi demonstrat pe H prin lanţurile de reguli 6,3, şi 8.

În practică există mai multe scheme de motoare de inferenţe, în funcţie de particularităţile ciclului de bază sau de modurile de înlănţuire a ciclurilor. Vom vedea acest lucru mai tîrziu.

3.3.2. Probleme funcţionale generale ale sistemelor expert

Într-un sistem expert regulile din baza de reguli sînt relativ independente, în situaţia în care nu se apelează direct unele pe altele. Comunicaţiile dintre reguli se realizează prin baza de fapte, în sensul că o regulă nu este declanşată decît atunci cînd anumite aspecte sînt recunoscute de către motorul de inferenţe, în situaţia curentă a bazei de fapte. **Situaţia curentă** a bazei de fapte este în permanenţă supravegheată. La fiecare ciclu EVALUARE-EXECUŢIE are loc o reevaluare continuă a stării sistemului precum şi o redistribuire a controlului. Cînd o operaţie din programul specific motorului de inferenţe se află în curs de execuţie, se spune că este în **control**. Redistribuirea controlului realizează lansarea în execuţie a unor noi operaţii din program.

Cunoaşterea operatorie poate fi îmbogăţită sau modificată prin manipulări locale limitate, întrucît regulile, în principiu, nu se apelează unele pe altele. Aceasta deschide posibilitatea **modularizării** cunoaşterii, în sensul că se pot **expanda, rafina sau transforma** regulile, prin intervenţia expertului, în funcţie de utilizările ce li se dau. Regulile nou introduse de către utilizatorii experţi sînt uşor integrate în baza de reguli. Modul de invocare a regulilor permite **crearea, modificarea, revizuirea, căutarea şi eliminarea** unei reguli, fără a ţine seama de celelalte existente deja.

Eventualele schimbări în baza de fapte pot influenţa rapid, la fiecare ciclu de bază al motorului de inferenţe, comportamentul global al sistemului expert. În această situaţie, se vorbeşte de **sensibilitatea** sau "reactivitatea" motorului de inferenţe, proprietate interesantă în aplicaţiile concrete. Sensibilitatea motorului de inferenţe creşte prin folosirea demonilor.

Reevaluarea continuă a stării sistemului expert (faza de filtrare) este o operaţie mare consumatoare de timp. De aceea, se caută îmbunătăţiri la nivelul organizării bazei de fapte, precum şi în mecanismele de căutare.

Redistribuirea controlului ridică problema dificilă şi importantă a REZOLVĂRII CONFLICTELOR. Utilizatorul expert, în redactarea părţii declanşator a regulilor, va exprima condiţiile relative la baza de fapte, circumstanţe care justifică sau interzic unele reguli, deja reţinută prin SELECŢIE sau introdusă în ansamblul de conflict, să fie supusă rezolvării conflictelor. Astfel, regulile care la un moment dat sînt colectate în ansamblul de conflict nu pot fi în situaţia de egalitate pentru obţinerea controlului. Lansarea în execuţie, cu prioritate, a primei reguli din ansamblul de conflict, este o proprietate intrinsecă a motorului de inferenţe. În asemenea caz, utilizatorul devine interesat să supravegheze ordinea în care el a declarat regulile. Problema se complică şi mai mult dacă utilizatorul expert foloseşte în scrierea regulilor **primitive de creare dinamică** incluse în **metareguli**, în scopul intervenţiei în derularea rezolvării conflictelor sau în selecţie, care nu vor mai fi implicite, aşa cum le-am prezentat anterior. Cu ajutorul primitivelor se realizează inhibarea sau validarea unor reguli, influenţîndu-le astfel posibilitatea execuţiei.

În funcţie de varietatea aplicaţiilor, **sistemele expert se diferenţiază între ele** prin tipul limbajului utilizat la reprezentarea cunoaşterii, prin particularităţile structurii şi prin modul de funcţionare. Problemele legate de aceste diferenţieri sînt în practică interdependente, cînd se doreşte conceperea unui sistem expert particular.

Probleme deosebite ridică **sistemele expert care reprezintă cunoaşterea incertă şi/imprecisă.** Astfel, **un fapt este incert** atunci cînd el reprezintă o aserţiune despre care nu se poate afirma că are valoarea T (true) sau F (false) şi, deci, poate să aibă oricare din cele două valori de adevăr. De exemplu, "Contul (X) poate fi cont de activ". Incertitudinea este introdusă prin expresia "poate fi".

O regulă este incertă dacă produce concluzii incerte, chiar dacă foloseşte premise certe. De exemplu:

DACĂ produsul-A are caracteristica-X
ŞI produsul-A are caracteristica-Y
ŞI produsul-A are caracteristica-Z
ATUNCI
este posibil ca produsul-A să fie achiziţionat.

Incertitudinea este introdusă prin expresia "este posibil".

Un fapt este imprecis dacă implică obiecte incomplet identificate. De exemplu, "Produsul-A are preţul de aproximativ 3.000 lei", conţine imprecizia introdusă prin cuvîntul "aproximativ".

O regulă este imprecisă dacă implică fapte imprecise în premisă sau în concluzie. De exemplu, "Dacă preţul produsului-A pe piaţa-1 este foarte mare, atunci achiziţionaţi de pe piaţa-2".

Notăm că teoria probabilităţilor dispune de modele de tratare a incertitudinii, inclusiv modele axiomatice care prezintă interes în dezvoltarea sistemelor expert. În aceeaşi măsură se folosesc teoria probabilităţilor şi teoria subansamblelor. În practică, limbajele de programare a inteligenţei artificiale pot trata incertitudinea şi imprecizia, cu ajutorul unor convenţii şi primitive prin care utilizatorul expert ataşează pieselor de cunoaştere un **grad de incertitudine** sau un **domeniu de imprecizie**.

Problema tratării incertitudinii şi impreciziei presupune îmbunătăţiri importante ale motoarelor de inferenţe, pentru derularea diferitelor faze ale ciclului EVALUARE-EXECUŢIE.

3.3.3. Tipologia motoarelor de inferenţă

Teoria motoarelor de inferenţe este poate cea mai atractivă şi mai complexă problematică din întreg ansamblul cunoştinţelor necesare dezvoltării sistemelor expert. Puţine lucrări tratează în mod expres această problematică. Încercînd o sinteză, care să orienteze cititorul în domeniul nostru de interes, nu putem să nu relevăm diferenţierile motoarelor de inferenţe şi tipologia acestora. Vom aborda tipologia motoarelor de inferenţe folosind **criteriile introduse de utilizatorul expert**, şi anume:

- modul de invocare a regulilor;
- metodele de inferenţă;
- strategiile de dezvoltare a căutării;
- monotonia şau non-monotonia;
- stilul de programare;
- limbajul de descriere a cunoaşterii.

i) Modul de invocare a regulilor se referă la faptul dacă motoarele de inferențe funcționează după strategia de control înainte, după strategia de control înapoi sau după strategia de control combinat (înainte-înapoi).

i1) Un motor de inferențe funcționează după strategia de control înainte dacă faptele din baza de fapte, asupra cărora se aplică declanșatorii regulilor, reprezintă piese de cunoaștere a căror valoare de adevăr este deja cunoscută (stabilită).

La acest tip de motoare, regulile se numesc "reguli-în-înainte" (forward rules sau antecedent rules). Exploatarea unor asemenea reguli corespunde la un raționament "începînd de la fapte spre scopuri, obiective". Avem de-a face cu ceea ce se numește în engleză forward reasoning sau facts oriented reasoning.

De exemplu, să considerăm secvența de reguli de mai jos:

[1] DACĂ cont-200 este cont-de-stocuri
ATUNCI cont-200 este cont-de-activ
[2] DACĂ operația este intrare-de-materiale
ȘI cont-200 este cont-de-activ
ATUNCI cont-200 se debitează
[3] DACĂ operația este de ieșire-de-materiale
ȘI cont-200 este cont-de-activ
ATUNCI cont-200 se creditează

În timpul funcționării motorului are loc compararea (pattern matching) condițiilor din reguli cu faptele din baza de fapte și dacă ele corespund, regula în cauză se va executa (fig.3.6.).

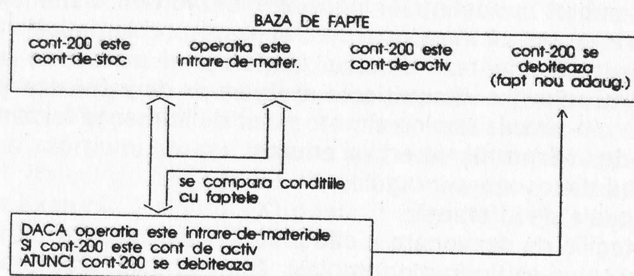


Figura 3.6. Execuția regulii modifică baza de fapte

Noile fapte adăugate în baza de fapte, pot fi și ele folosite în continuare pentru noi raționamente, de același tip, prin înlănțuirea regulilor "în-înainte". Se ajunge la un lanț de inferențe format prin execuția succesivă a regulilor, ca în fig.3.7.

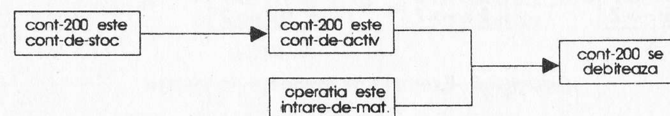


Figura 3.7. Lanț de inferență în-înainte pentru deducția debitării contului 200

Considerăm că explicăm mai bine această metodă de invocare "în-înainte" a regulilor, dacă prezentăm și un exemplu în care atît faptele cît și regulile sînt notate simbolic (fig. 3.8.).

Exemplul din fig. 3.8. detaliază acest raționament în dinamica sa, urmărind să deducă pe Z dacă F și B există în baza de fapte. Raționamentul se desfășoară astfel:

- În fiecare moment se compară setul de fapte cu declanșatorii din reguli. O regulă se execută numai o singură dată, chiar dacă se verifică această împerechere ori de cîte ori este necesar;
- Prima regulă care se execută este A → D, deoarece A este deja prezent în baza de fapte. Execuția acestei reguli determină introducerea lui D în baza de fapte;
- Noua bază de fapte face posibilă execuția regulii C & D → F, care va introduce pe F, rezultînd altă configurație a bazei de fapte;
- Această nouă bază de fapte determină execuția celei de a treia reguli F & B → Z, care va introduce pe Z, obținîndu-se ultima bază de fapte. Se observă din fig. 3.9. cum căutarea unor noi fapte se face numai în direcția săgeții, care separă cele două părți ale regulilor (declanșatorul din stînga și acțiunile din dreapta).

În principiu, strategia de control înainte este mare consumatoare de resurse și mai ales scumpă din punct de vedere al costului. Mai eficientă sub aspectul costului este invocarea regulilor "în-înapoi", specifică strategiei de control cu același nume.

i2) Un motor de inferențe funcționează după strategia de control înapoi atunci cînd:

- anumite fapte din baza de fapte trebuie dovedite sau stabilite și se numesc "ipoteze" sau "probleme de rezolvat" sau "scopuri";

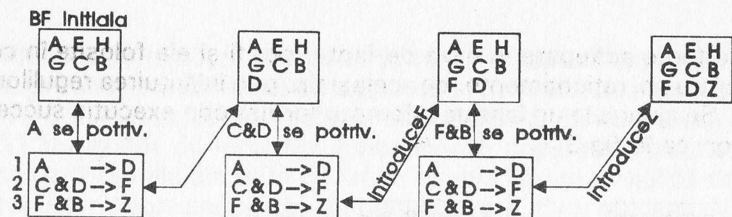


Figura 3.8. Exemplu de invocare în-înainte

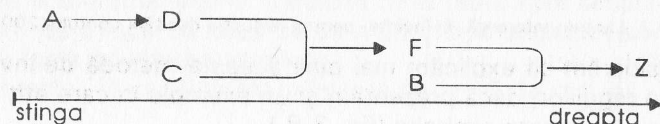


Figura 3.9. Lanț inferențial produs de invocarea în-înainte, conform figurii 3.8.

- declanșatorii regulilor se referă numai la faptele de dovedit;

- cînd o regulă este declanșată, noile fapte definite prin acțiunile regulilor pot fi introduse în baza de fapte ca noi fapte de dovedit;

- problemele de rezolvat la care se referă declanșatorii sînt considerate soluționate dacă s-au dovedit că sînt probleme declarate apriori ca "primitive", ori au fost soluționate anterior; numai în această situație acțiunea descrisă de regulă se va executa.

La acest tip de motoare, regulile se numesc "reguli-în-înapoi" (backward rules sau consequent rules).

Execuția acțiunii, reprezentată de o regulă-în-înapoi nu se realizează decît după declanșare și nu în același timp ca la strategia de control înainte. Exploatarea unor asemenea reguli corespunde raționamentului invers "de la scop către fapte" numit în engleză backward reasoning sau goal oriented reasoning.

În fig. 3.10. prezentăm schema de lucru pentru acest mod de invocare a regulilor. Vom lucra cu aceeași bază de cunoștințe ca și în exemplul anterior.

În pasul 1, motorul stabilește că trebuie dovedit Z, apoi decide că trebuie să stabilească pe F și pe B, în scopul de a conchide pe Z.

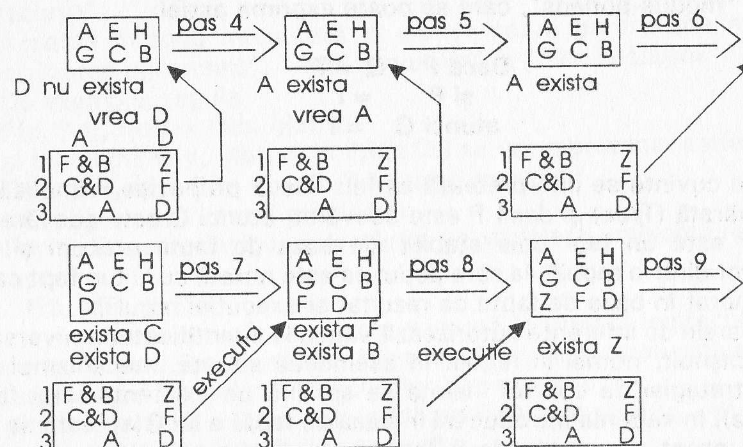


Figura 3.10. Lanț de inferențe pentru invocarea regulilor în-înapoi

În pasul 2, se încearcă stabilirea lui F și se găsește o regulă care conchide pe F, de la care decide că trebuie stabilite C și D.

În pașii 3-5 este găsit C și decide că trebuie stabilit A înainte de a conchide pe D; va găsi pe A în baza de fapte.

În pașii 6-8 se execută regula 3 pentru stabilirea lui D, după care se execută regula 2 pentru stabilirea lui F, iar în final regula 1 pentru stabilirea lui Z - tocmai scopul inițial.

Deosebirea față de lanțul inferențial de la invocarea regulilor "în-înainte" constă în modul în care faptele și regulile sînt căutate.

i3) Un motor de inferențe funcționează după strategia de control mixtă (combinată, înainte-înapoi) sau **bidirecțională**, dacă o parte a faptelor din baza de fapte sînt considerate "de-stabilit" sau de dovedit (sînt probleme sau ipoteze) iar altele sînt considerate stabilite sau dovedite deja. Declanșatorii regulilor pot fi lansați simultan pe fapte de un tip sau altul.

ii) Metodele de inferență sînt foarte diverse și anume:

- modus-ponens;
- modus-ponens-extins;
- specializarea universală;

- modus-tollens;
- principiu de rezoluție.

Stategia de control înainte corespunde, în sens restrîns metodei de inferență "modus-ponens", care se poate exprima astfel:

Dacă P → Q = T
și P = T
atunci Q = T

ceea ce în cuvinte se interpretează astfel: "Dacă propoziția P implică Q este adevărată (True) și dacă P este adevărat, atunci Q este adevărat"; în care P este un fapt deja stabilit din baza de fapte precum și un declanșator dintr-o regulă, la care acțiunea este notată cu Q (un fapt care va fi memorat în baza de fapte ca rezultat al execuției regulii).

Motoarele de inferențe autorizează variabile **cuantificatori universali**, în mod obișnuit, numai în reguli. În asemenea situații, mecanismul de bază al strategiei de control înainte se sprijină pe existența unui fapt stabilit P(a), în vederea introducerii în baza de fapte a lui Q(a), care se va stabili. În acest caz avem de a face cu un motor de inferențe care combină metoda "modus-ponens" cu metoda de inferență numită "specializare universală". Conform metodei **"specializare universală"**,

Dacă R(x) este adevărat, oricare ar fi (x),
atunci R(a) este adevărat.

În aceste condiții, prin specializare se poate afirma că:

Dacă P(x) → Q(x) = T, oricare ar fi (x),
atunci P(a) → Q(a) = T,

de unde prin "modus-ponens" conchidem că întrucît P(a) este adevărat și Q(a) este adevărat.

Mecanismul de bază al strategiei de control înapoi permite introducerea unei probleme P (fapt de stabilit), din moment ce există o regulă cu declanșatorul Q și acțiunea P.

Motoarele de inferență admit variabile în regulile care folosesc specializarea universală, pentru a invoca o regulă al cărei declanșator este de forma Q(x), dacă există o problemă Q(a). În această situație, în momentul în care o problemă P introdusă anterior, prin invocare "înînapoi", printr-o regulă R (care are pe Q declanșator și P ca acțiune), a fost rezolvată, se pot aplica "specializarea universală" și "modus-ponens", pornind de la regula R, pentru a conchide că Q este stabilită (faptul care ar putea fi propagat prin specializare și "modus-ponens",

plecînd de la regula care a introdus pe Q ș.a.m.d.).

Motoarele de inferențe construite în PROLOG admit variabile atît în reguli cît și în fapte, prin utilizarea metodei de inferență numită **"principiu de rezoluție"**.

Această metodă integrează și generalizează printre alte metode, "specializarea universală", "modus-ponens" și "modus-tollens".

De exemplu, regula
P(x, f(a), f(u)) → Q(b, g(x), u)
și problema **Q(y, g(b), c)** în PROLOG se vor reprezenta astfel:
Q(b, g(x), u): - P(x, f(a), f(u)) în cazul regulii și
:-Q(y, g(b), c) în cazul problemei.

Prin aplicarea principiului rezoluției se va infera problema reprezentată astfel:
:-P(b, f(a), f(c)).

La motoarele de inferențe care tratează informații imprecise (vagi) și/incerte, metodele de inferență de bază, specializarea universală și modus-ponens, fac obiectul extensiilor. De exemplu:

Dacă regula "P → Q nu este certă decît într-un grad g_1", și dacă "P nu este certă decît într-un grad g_2",
atunci se va putea infera potrivit acestei reguli că Q este cert în gradul g_1*g_2"

Din literatură s-a desprins și un alt exemplu:

Dacă un motor tratează fapte și reguli imprecise plecînd de la regula certă "P → Q" în care P și Q sînt imprecise iar P_1 este aproape de P_2, se poate infera că "Q_1este apropiat de Q_2".

De obicei, în sistemele expert care tratează cunoștințe incerte și/imprecise, mecanismele inferențiale nu se limitează numai la punerea în lucru a fiecărei reguli, independent de rezultatele anterioare. Se utilizează și alte metode de inferență în vederea agregării rezultatelor intermediare, prin exploatarea mai multor reguli. De exemplu, **metoda "modus-ponens-extinsă"** acționează astfel:

Fie o regulă R_1, care atribuie unui fapt F un grad de certitudine g_{c1} și o altă regulă R_2, care atribuie aceluiași fapt un grad de certitudine g_{c2}, unde $0 \leq g_{c1}$, și $g_{c2} \leq 1$. În această situație, se va folosi modus-ponens-extins, care determină **atașarea la F a unui grad de certitudine combinat,** astfel:

$$g_{c1} - g_{c2} - g_{c1} * g_{c2}$$

În sistemele expert cu motoare de inferenţe care tratează imprecizia ataşată la fapte şi/sau la reguli, se extinde "specializarea universală" căutîndu-se modele de compatibilitate între declanşatorii din reguli şi fapte. De exemplu, în condiţiile regulii "în-înainte" de mai jos:

Dacă PIAŢA ESTE SATURATĂ
atunci REDUCEM PRODUCŢIA

şi a faptului:

PIAŢA ESTE FOARTE SATURATĂ

nu există nici un dubiu că regula trebuie să se declanşeze şi, deci, să fie reţinută în ansamblul de conflict (R_2). Dar tot aşa de indubitabil trebuie ca, în faza rezolvării conflictelor, să se ţină seama de un grad mai mic sau mai mare de compatibilitate între fapte şi reguli. Dacă regula este, totuşi declanşată, atunci trebuie să se ţină seama de gradul de compatibilitate care a determinat declanşarea şi să fie ajustată, în mod adecvat concluzia.

iii) Strategiile de dezvoltare a căutării. Pe modurile de invocare a regulilor într-un motor de inferenţe se suprapun, de fapt, şi unele dintre strategiile clasice de dezvoltare a căutării în spaţiul stărilor şi în spaţiul subproblemelor: căutarea în adîncime (depth first search), căutarea în lăţime (pe nivel) şi căutarea ordonată.

iii1) Căutarea în spaţiul stărilor. Spaţiul stărilor, corespunzător unei probleme particulare, se defineşte prin aplicarea tuturor regulilor în toate modurile posibile, începînd cu starea iniţială, care este baza de fapte iniţială.

Spaţiul stărilor se reprezintă printr-un graf orientat, în care fiecare vîrf S_i reprezintă o stare şi fiecare arc S_iS_j reprezintă aplicaţia unei reguli care transformă starea S_i a bazei de fapte în starea S_j. Starea obiectiv, care corespunde succesului căutării, se defineşte prin una din convenţiile:

a) o bază de fapte în care apare un fapt care satisface anumite caracteristici;

b) o bază de fapte în care nu mai există fapte de demonstrat, de stabilit, de dovedit (are loc o "oprire prin saturaţie");

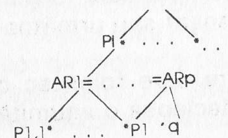
c) o bază de fapte pe care nici o regulă nu o mai poate modifica.

Convenţiile a) şi c) se utilizează la motoarele care funcţionează pe baza strategiei de control înainte, iar convenţia de la punctul b) este

folosită la motoarele care funcţionează după strategia de control înapoi.

iii2) Căutarea în spaţiul subproblemelor. Spaţiul subproblemelor se defineşte prin aplicarea regulilor ca operatori de descompunere a problemei în toate modurile posibile, plecînd de la problema iniţială.

Spaţiul subproblemelor se reprezintă printr-un graf orientat de tip ŞI-SAU (vezi fig.3.11), în care fiecare vîrf simbolizează fie o problemă P_i (P_i este vîrf problemă), fie o aplicaţie AR_j a unei reguli (AR_j este vîrf regulă). Dacă AR_1... AR_p sînt toate aplicaţiile regulilor care descompun problema P_i, atunci ansamblul arcelor P_iAR_j, pentru un i fixat de la 1 la p şi un j de la 1 la p, determină o legătură "SAU". Dacă P'_1... P'_q sînt toate subproblemele lui P_i introduse prin AR_j atunci ansamblul arcelor AR_jP_i,penttru un i fixat de la 1 la q, determină o legătură "ŞI".



Aplicaţia regulii AR1 descompune problema Pi în subproblemele P1,1... Pi,q. Dacă ambele probleme vor fi considerate "rezolvat", atunci şi Pi va fi rezolvată

Figura 3.11. Grafic de descompunere a problemei Pi

Se observă că, concluzia unei reguli rezultă prin conjuncţia (aplicarea operatorului "ŞI") faptelor specificate. O concluzie este dedusă prin aplicarea unei succesiuni de reguli (disjuncţie) cu operatorul "SAU". În vîrfurile "ŞI", toate arcele trebuie să fie adevărate pentru ca însuşi vîrful să fie adevărat. În vîrfurile "SAU" un singur fapt trebuie să fie adevărat. Vîrfurile "ŞI" sînt marcate prin arc. Arcul "SAU" este o regulă simplă, cu o singură condiţie. Vîrful "ŞI" reprezintă o regulă şi fiecare arc este o condiţie.

Se spune că fiecare vîrf-problemă decompozabil este tată pentru vîrfurile reguli, iar fiecare vîrf-regulă este tată pentru vîrfurile probleme.

Starea obiectiv, într-o asemenea reprezentare, poate fi atinsă dacă, după exploatarea (căutarea) în spaţiul subproblemelor, se poate afirma că problema iniţială a fost rezolvată în **condiţiile**:

- un vîrf-problemă reprezentativ este, prin definiţie, rezolvat;
- un vîrf-problemă are un fiu (vîrf-regulă), de asemenea rezolvat;
- un vîrf-regulă la care toţi fii (vîrfurile-probleme) sînt considerate "rezolvat".

MICROMOTOR din exemplul anterior este construit pe principiul strategiei de dezvoltare a căutării în adîncime.

Strategiile de dezvoltare a căutării joacă un rol cheie în definirea fazelor de SELECȚIE și / REZOLVARE A CONFLICTELOR din ciclul de bază al motoarelor de inferență. De exemplu, dacă motorul funcționează după strategia de control înapoi, numai așa-zisele probleme (fapte de demonstrat) se vor compara cu declanșatorii regulilor în faza de FILTRARE, iar subansamblul F1 se va compune numai din asemenea fapte. Dacă motorul va urma strategia de căutare în adîncime, în F1 se vor găsi numai faptele de stabilit rezultate din ultima declanșare a regulii, în strictă ordine a declanșatorilor regulii. Faza de REZOLVARE A CONFLICTELOR este deasemenea dependentă de strategia de căutare, care poate comanda o "întoarcere-înapoi" și o repunere în acțiune a declanșării unei reguli reținute în faza de FILTRARE. Uneori, ansamblul de conflict este format în prealabil, dar se alege arbitrar prima regulă de declanșat sau următoarea, în cazul unei strategii de control înapoi.

Anumite sisteme expert au motoare de inferențe care folosesc o funcție euristică intrinsecă, bazată pe interesul de a declanșa o anumită regulă.

Alte sisteme sînt dotate cu "metareguli", furnizate de către utilizatorul expert, pentru administrarea într-o anumită manieră a regulilor. Numeroase sisteme expert urmează **regimul de control irevocabil**, în care declanșarea regulilor alese la rezolvarea conflictelor nu este niciodată influențată de utilizator (pusă în discuție). Din acest punct de vedere, se poate vorbi de motoare de inferențe care funcționează după strategia de căutare "în adîncime", "în lățime", sau "ordonată și irevocabilă".

Există motoare de inferență care urmează **regimul de control prin tentative**, conform căruia încercarea aplicării unei reguli sau ansamblu de reguli este urmată de punerea în cauză a acestei încercări și întoarcerea controlului înapoi (backtracking). Din această cauză, se spune că o strategie poate fi "în adîncime", "în lățime", "ordonată" și "prin tentative"[1].

În conformitate cu strategia de dezvoltare a căutării utilizată, un motor de inferențe poate intra în așa-numita "buclare" (ciclare continuă), stabilind sau nu rezultatul așteptat, dar consumînd resurse (timp și

---

[1] Nilsson folosește expresia "backtracking strategy" în cazul strategiilor simultan prin tentative și în adîncime

memorie) foarte importante.

Mai ușor de programat se consideră strategia de căutare în adîncime, deoarece se poate respecta o anumită limită a adîncimii, în vederea evitării buclelor.

iv) Monotonia și non-monotonia. Monotonia și non-monotonia sînt proprietăți ale sistemelor expert rezultate din funcționarea motoarelor de inferențe.

iv1) Monotonia se referă la modul de funcționare a motorului de inferențe și se consideră că un motor de inferențe funcționează "monoton" sau este monoton dacă:

- nici o piesă de cunoaștere (fapt sau regulă) nu poate fi eliminată din baza de cunoștințe și
- nici o piesă de cunoaștere nou adăugată în baza de cunoștințe nu introduce contradicția.

Asemenea motoare funcționează conform principiului demonstratoarelor de teoreme obișnuite în logica clasică (logica propozițiilor și logica predicatelor), în sensul că se pleacă de la axiome (fapte sau reguli) și, prin aplicarea metodelor de inferență, autorizează sau inferă noi formule (fapte sau reguli) care se adaugă la cele existente, fără eliminarea altora.

La motoarele monotone, ordinea de declanșare a regulilor nu influențează structura bazei de fapte finale.

Monotonia se recomandă în cazul sistemelor expert pentru diagnostic, care pot acorda consultații plecînd de la ansamblul simptomelor observate, fără a ține seama de evoluția în timp a situației.

iv2) Non-monotonia este atunci cînd, în timpul funcționării, motorul de inferențe permite eliminarea sau modificarea unor piese de cunoaștere din baza de cunoștințe.

Există trei tipuri de situații care obligă la construirea motoarelor non-monotone:

- situațiile diagnosticului interactiv, în care pot interveni decizii asupra unui fapt observat;
- situațiile în care se încearcă mai multe linii de raționament pentru construirea ipotezelor. De exemplu, se încearcă generarea unui plan de acțiuni (sub forma faptelor provizoriu stabilite), a efectelor presupuse pentru fiecare acțiune inclusă în plan, iar pentru a putea continua planificarea trebuie mereu revăzute, reconsiderate acțiunile prevăzute deja, împreună cu efectele asociate, prin modificarea sau eliminarea lor;
- situațiile în care se folosesc reguli implicite (plauzibile prin lipsă), ale căror concluzii fac dinainte subiectul revizuirii (valabile și în diagnostic și

în planificare). De exemplu, regula:

"Dacă PRODUSUL-A NU SE VINDE PE PIAȚA-1
atunci PIAȚA-1 ESTE SATURATĂ"

poate fi considerată ca o regulă implicită atunci cînd concluzia nu este certă și anume:

"Dacă PRODUSUL-A NU SE VINDE PE PIAȚA-1
atunci (în absența altor informații, contradictorii)
PIAȚA-1 ESTE SATURATĂ".

Dacă, ulterior, au devenit mai evidente cauzele pentru care nu se vinde produsul-A, atunci concluzia care s-a tras anterior, "PIAȚA-1 ESTE SATURATĂ", va fi reconsiderată.

Credem că trebuie să remarcăm, în acest punct al expunerii, că limbajul PROLOG, funcționează el însuși ca un motor de inferențe pe baza strategiei de control înapoi, a strategiei de căutare în adîncime, în regim prin tentative și este capabil de non-monotonie, permițînd variabile și în fapte și în reguli. Din acest punct de vedere, oricare programe PROLOG pot fi considerate sisteme expert.

v) Stilul de programare. Stilul de programare se referă la punerea la punct și revizuirea progresivă a bazei de cunoștințe, în manieră declarativă sau incrementală, conform principiilor dezvoltării sistemelor expert.

Stilul de programare declarativă sau incrementală oferă posibilitatea ca mai mulți experți să poată contribui la actualizarea bazei de cunoștințe în mod autonom și relativ independent. Se obține în felul acesta un cîștig în flexibilitate rezultat din principiile de organizare și funcționare a sistemelor expert. În aceste condiții, cunoștințele transmise de experți trebuie să respecte convențiile pentru metodele de inferență și strategiile implementate în motoare și, în plus, să determine aptitudini de îmbunătățire a activității motoarelor de inferențe.

Trebuie știut că, în principiu, din punctul de vedere al tratării regulilor, motoarele de inferențe sînt insensibile la ordinea regulilor în baza de reguli, au o relativă independență față de cunoștințele adăugate ulterior și, mai ales, față de reguli, întrucît pot apare incoerențe logice.

vi) Limbajul de descriere a cunoașterii. Limbajul de descriere a cunoașterii este specific atît bazei de reguli, cît și bazei de fapte. Expertul transmite cunoașterea sa unui specialist în sisteme expert, care-l intervievează. Această cunoaștere trebuie descrisă printr-un limbaj specializat ca să poată fi utilizată de către programul de inteligență artificială.

Limbajul de descriere este util în proiectarea bazei de cunoștințe, motiv pentru care trebuie să fie general și să nu influențeze conținutul cunoașterii transmise de expert. De gradul de generalitate al limbajului de descriere a cunoașterii depinde și puterea motorului de inferențe care prelucrează cunoașterea.

În funcție de limbajul de descriere a cunoașterii pot exista:
- motoare de ordin zero (0);
- motoare de ordin zero plus (0+);
- motoare de ordin unu (1).

La motoarele de ordin zero, regula se descrie folosind premize și concluzii relative la o propoziție elementară, confirmată sau infirmată, ca mai jos:

```
⟨regula⟩    ::= DACĂ ⟨premise⟩ ATUNCI ⟨concluzii⟩
⟨premise⟩   ::= ⟨premisă⟩ | ⟨premisă⟩ ȘI ⟨premise⟩
⟨premisă⟩   ::= ⟨propoziție⟩ | NOT ⟨propoziție⟩
⟨concluzii⟩ ::= ⟨concluzie⟩ | ⟨concluzie⟩ ȘI ⟨concluzii⟩
⟨concluzie⟩ ::= ⟨propoziție⟩ | NOT ⟨propoziție⟩
⟨fapt⟩      ::= ⟨propoziție⟩ | NOT ⟨propoziție⟩
⟨scop⟩      ::= ⟨propoziție⟩ | NOT ⟨propoziție⟩?
```

La motoarele de ordin zero plus (0+), regulile prezintă particularitatea de a utiliza atribute în descrierile lor. Un atribut al unei probleme este o proprietate caracteristică, ce poate lua una sau mai multe valori. De exemplu, pentru un salariat, starea civilă este un atribut care poate avea următoarele valori posibile: necăsătorit, căsătorit, divorțat, văduv.

Premisele unor asemenea reguli permit compararea atributelor între ele sau a atributelor cu valorile. De exemplu, în regula următoare:

DACĂ vîrsta < 18
ATUNCI statut ← minor

Se poate constata că prezența faptului "vîrsta = 15" în baza de fapte nu este suficientă pentru validarea premisei acestei reguli, deoarece trebuie testată condiția "15 < 18". Validarea unei premise de acest tip presupune căutarea informației corespunzătoare în baza de fapte pentru realizarea comparației.

Concluziile afectează valoarile atributelor. De exemplu, în regula de mai jos:

DACĂ salariatul este căsătorit
ATUNCI starea civilă ← căsătorit

Trebuie făcută diferențierea între atributele care iau o singură valoare și atributele care iau valori multiple. De exemplu un salariat poate avea mai mulți copii și se va nota: copii = Paul, copii = Ion, copii = Maria etc. Un asemenea atribut se numește "multivaloare" în raport cu atributele "monovaloare".

În condițiile acestor precizări, elementele structurale dintr-o regulă se vor descrie în conformitate cu limbajul de descriere următor:

<premisă> ::= <atribut> <comparator> <valoare>
<premisă> ::= <atribut> <comparator> <atribut>
<comparator> ::= < | > | = | = | < | >
<atribut> ::= <atribut monovaloare> | <atribut multivaloare>
<concluzie> ::= <atribut> <simbol-de-atribuire> <valoare>
<concluzie> ::= <atribut> <simbol-de-atribuire> <atribut>
<simbol-de-atribuire> ::= <simbol-de-atribuire-monovaloare> |
<simbol-de-atribuire-multivaloare>
<simbol-de-atribuire-monovaloare> ::= ←
<simbol-de-atribuire-multivaloare> ::= ⇐
<fapt> ::= <atribut> = <valoare>
<scop> ::= <atribut> = ?|<atribut> = <valoare> ?

La motoarele de ordin unu (1), limbajul de descriere autorizează folosirea variabilelor. Regulile descrise în asemenea limbaj au un caracter mai general decît regulile de la motoarele de ordin 0 sau de ordin 0+.

O astfel de regulă poate, pentru o anumită structură a bazei de fapte, să se declanșeze de mai multe ori, conform diferitelor valori ale variabilelor. De exemplu, să considerăm regula:

DACĂ tatăl (X) = Y	**Y este tatăl lui X**
ȘI frate (y) = F	**F este fratele lui Y**
ATUNCI UNCHI (X) ⇐ Y	**Y este unchiul lui X**
ȘI sex (Y) ← masculin	**Y este de sex masculin**
Și sex (F) ← masculin	**F este de sex masculin**

În condițiile în care baza de fapte conține faptele:

tatăl (PAUL) = Ion
frate (Ion) = Radu
frate (Ion) = Petru

regula de mai sus se va declanșa succesiv, pentru valorile:

X = Paul ; Y = Ion ; F = Radu;
X = Paul ; Y = Ion ; F = Petru.

După aceste declanșări, baza de fapte va conține în plus noi fapte, astfel:

unchi (Paul) = Petru ; unchi (Paul) = Radu
sex (Petru) = masculin; sex (Radu) = masculin

În aceste condiții, regulile se descriu în conformitate cu limbajul SNARK (Symbolic Normalized Acquisition and Representation Knowledge):[1]

<premisa> ::= <relație> <obiect> <comparator> <obiect>
<premisa> ::= <obiect> <comparator> <obiect>
<obiect> ::= <valoare> | <variabila>
<valoare> ::= <numar> | <constanta-literal>
<concluzie> ::= <relație> < obiect> <atribuire> <obiect>
<concluzie> ::= <actiune>
<atribuire> ::= <atribuire-monovaloare> | <atribuire-multivaloare>
<atribuire-monovaloare> ::= ←
<atribuire-multivaloare> ::= ⇐
<fapt> ::= <relație> <valoare> = <valoare>
<scop> ::= <relație> <valoare> = ?
<scop> ::= <relație> <valoare> = <valoare> ?

Se observă introducerea noțiunii de relație, cunoscută sub numele de "relație binară". De exemplu, pentru propoziția:

---

[1] Laurière, J.L., "Représentation et utilisation des connaissances," T.S.I. 1 et 2, 1982, p.25-42, 109-133; Vezi și Rousset, M.C., TANGO: moteur d'inferènces pour une classe de systèmes experts avec variables, Thèse de 3eme Cycle, Orsy, 1983.

"Creşterea masei monetare este sursa inflaţiei"
se poate scrie relaţia:

CAUZA (creşterea masei monetare, inflaţie), care favorizează răspunsul la astfel de cereri:

"Cauzele inflaţiei?"
Consecinţele creşterii masei monetare?"

* *
*

Motoarele de inferenţe se diferenţiază între ele şi din punctul de vedere al utilizatorilor obişnuiţi, în sensul că pot exista motoare care funcţionează interactiv sau nu, motoare care folosesc un anumit limbaj de comunicaţie cu utilizatorul, ori care sînt dotate sau nu cu un modul de explicaţie a propriilor inferenţe, etc.

3.3.4. Scheme de bază pentru motoare de inferenţe

Prezentarea unor scheme pentru motoarele de inferenţe, cu funcţionare conform strategieide control înainte, strategiei de control înapoi, strategiei de căutare în adîncime sau în lăţime, cu regim irevocabil sau prin tentative, cu sau fără monotonie, cu sau fără variabile, precum şi a celor care permit planificarea acţiunilor, asigură o bază solidă pentru trecerea la studiul şi dezvoltarea sistemelor expert complexe în domeniul particular al contabilităţii.

Se înţelege că motoarele de inferenţe ale multor sisteme expert pentru aplicaţii efective sînt mult mai sofisticate decît schemele prezentate în acest material şi nu ne-ar oferi posibilitatea, într-un spaţiu atît de redus, să le prezentăm în detaliile lor funcţionale. De aceea, am considerat mai util să dăm un grupaj de scheme, redactate într-un limbaj simbolic simplificat, uşurînd astfel înţelegerea lor şi oferind o încurajare pentru dezvoltări noi.

Vom folosi, în continuare, un cod simplu de scriere a programelor în care am respectat convenţiile:

- săgeata indică atribuirea unor valori variabilelor:
- Dacă ... atunci ... sînt structuri alternative cunoscute din programarea clasică, completate eventual cu apeluri recursive şi proceduri;
- cuvintele cheie sînt subliniate. Cuvîntul cheie "restituie" determină returnarea unei valori, ca urmare a evaluării expresiei care îi urmează (poate fi ea însăşi un apel de procedură = recursivitate), şi abandonarea

procedurii în cauză;

- variabilele din antetul definiţiilor de proceduri sînt "protejate" (în sensul că vor fi rezolvate înaintea apelului procedurilor) şi restaurate. Deci, variabilele sînt "locale" fiecărei proceduri. Variabilele care apar în corpul procedurilor sînt "globale". S-au notat cu majuscule variabilele care primesc o valoare în momentul apelului, şi cu litere mici variabilele auxiliare, care nu primesc asemenea valori. Variabilele din corpul procedurilor sînt notate cu majuscule.

Funcţionarea motoarelor se va exemplifica pe baza de cunoştinţe[1] din fig.3.12.

Baza de reguli (BR)

1	K,	L,	M	-->	I
2	I,	L,	J	-->	Q
3	C,	D,	E	-->	B
4	A,	B,		-->	Q
5	L,	N, O,	P	-->	Q
6	C,	H,		-->	R
7	R,	J,	M	-->	S
8	F,	H		-->	B
9	G			-->	F

Baza de fapte (BF)

A , C , D , E , G , H , K

Figura 3.12. Baza de cunoştinţe simbolice[1].
În stînga regulii se află declanşatorul iar în dreapta se află acţiunea.

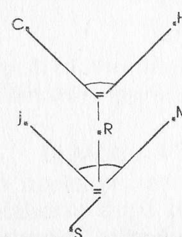


Figura 3.13. Graf ŞI-SAU pentru înlănţuirea regulilor 6 şi 7. (Dacă C şi H sînt stabilite, atunci R este stabilit. Dacă R şi J şi M sînt stabilite, atunci S este stabilit). "=" reprezintă condiţia, iar "*" reprezintă acţiunea sau producţia

Interpretarea se face astfel: "Dacă K şi L şi M sînt fapte stabilite, atunci se conchide că I este stabilit". Înlănţuirile regulilor se pot reprezenta printr-un graf ŞI-SAU ca în fig.3.13.

[1] Am adaptat exemplele din Farreny, H., Op. cit., p.79 ş.u.

3.3.4.1. Scheme de motoare bazate pe strategia de control înainte, cu regim irevocabil și monotonie

Toate convențiile menționate mai sus trebuie avute în vedere.

Vom prezenta mai întîi o schemă pentru un motor de inferențe cu integrarea imediată a concluziilor și apoi o schemă demotor tot cu invocarea regulilor "în-înainte", dar cu strategie de căutare "în lățime".

a) Motorul "STINA-1", cu integrare imediată a concluziei regulilor în baza de fapte, este dat în fig.3.14.

În această figură, cele două proceduri constituie o schemă de motor de inferențe care poate fi lansat prin apelul procedurii STAB-UN-FAPT. De exemplu, pentru stabilirea lui Q se va apela procedura STAB-UN-FAPT ("Q").

procedura STAB-UN-FAPT (FAPT)

1 dacă FAPT aparține lui BF, atunci restituie "succes"
2 restituie EXECUTA-UN-CICLU (BR)

procedura EXECUTA-UN-CICLU (REGULI, oregula)

1 dacă REGULI este vidă, atunci restituie "eșec"
2 OREGULA <--- alege o regulă oarecare din REGULI (de exemplu prima întîlnită)
3 REGULI <--- REGULI mai puțin OREGULA
4 dacă toate faptele din declanșatorul lui OREGULA există lui BR, atunci
4.1 început
4.2 dacă acțiunea lui OREGULA este FAPT, atunci restituie "succes"
4.3 adaugă acțiunea lui OREGULA în BF (dacă nu este deja în BF)
4.4 BR <--- BR mai puțin OREGULA
4.5 restituie EXECUTA-UN-CICLU (BR)
4.6 sfîrșit
5 restituie EXECUTA-UN-CICLU (REGULI)

Figura 3.14. Schema motorului de inferențe "STINA-1", cu invocarea regulilor "în-înainte" și introducerea imediată a faptului stabilit prin acțiunea regulii în BF

Motorul "STINA-1", invocă regulile după strategia de control înainte, în sensul că face comparații (pattern matching) ale declanșatorilor din reguli cu piesele de cunoaștere din baza de fapte, considerînd fiecare fapt ca fiind stabilit. Cînd o regulă este declanșată, concluzia este imediat stabilită iar faptul respectiv, pe care-l conține, este introdus în baza de fapte. În timpul funcționării motorului, baza de fapte BF conține numai fapte stabilite. De exemplu, funcționarea motorului STINA-1, pentru stabilirea faptului "Q", determină lansarea procedurii STAB-UN-FAPT ("Q"), astfel:

- se delanșează regula 3 din BR, pentru a adăuga faptul "B" în BF, apoi regula 4 pentru adăugarea lui "Q" iar apelul inițial al procedurii se termină prin restituirea valorii "succes". Înlănțuirea regulilor 3 și 4 s-a efectuat ca în graful SI-SAU din fig.3.15., care este un "graf de subprobleme" iar baza de fapte s-a modificat ca în "graful de stări" din fig.3.16.

Ciclul de bază pentru motorul STINA-1 s-a desfășurat astfel:

- în faza de SELECȚIE s-a eliminat regula care a fost deja executată (vezi instrucțiunea 4.4. din procedura EXECUTA-UN-CICLU, fig.3.14);
- în fazele de FILTRARE și REZOLVARE A CONFLICTELOR se execută instrucțiunile 1-4 ale aceleași proceduri.

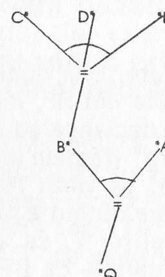


Fig. 3.15. Graf de subprobleme privind înlănțuirea regulilor 3 și 4

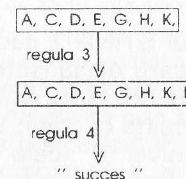


Fig. 3.16. Graf de stări privind STINA - 1 pentru stabilirea faptului "Q"

b) Motorul "STINA-2", cu invocarea regulilor "în-înainte", se lansează în aceleași condiții ca STINA-1, pentru stabilirea faptului "Q", dar este elaborat după o schemă conformă strategiei de căutare "în lățime" sau pe nivel (vezi fig.3.17.)

procedura STAB-UN-FAPT (FAPT, faptenoi)

1 dacă FAPT face parte din BF, atunci restituie "succes";
2 FAPTENOI ← listă vidă
3 restituie EXECUTĂ-UN-CICLU (BR, FAPTENOI)

procedura EXECUTĂ-UN-CICLU (REGULI, FAPTENOI, oregula)

1 dacă OREGULA este vidă, atunci
1.1 început
1.2 dacă FAPTENOI este vidă, atunci restituie "eșec"
1.3 BF ← BF plus FAPTENOI

1.4 FAPTENOI ← listă vidă
1.5 restituie EXECUTĂ-UN-CICLU (BR, FAPTENOI)
1.6 sfîrșit
2 OREGULA ← alege o regulă oarecare din REGULI (de exemplu, prima întîlnită)
3 REGULI ← REGULI mai puțin OREGULA
4 dacă toate faptele din declanșatorul lui OREGULA există în BF, atunci
4.1 început
4.2 dacă concluzia din OREGULA este FAPT, atunci restituie "succes"
4.3 adaugă concluzia lui OREGULA în FAPTENOI (dacă nu este deja în BF sau în FAPTENOI)
4.4 BR ← BR mai puțin OREGULA
4.5 sfîrșit
5 restituie EXECUTĂ-UN-CICLU (REGULI, FAPTENOI)

Figura 3.17. Schema motorului STINA-2, invocă reguli "în-înainte" și produce fapte prin căutare "în lățime"

Motorul STINA-2 caută să declanșeze cu prioritate, una după alta, regulile la care declanșatorii sînt compatibili cu faptele din BF, înaintea adăugării de noi fapte în BF, și de a le utiliza pentru declanșarea de noi reguli. În aceste condiții, vor exista "fapte de nivel zero" (faptele inițiale), "fapte de nivel 1" (cele obținute din faptele inițiale, pe baza regulilor aplicate asupra lor) ș.a.m.d., "fapte de nivel n + 1" (cele obținute pe baza faptelor de nivel n). STINA-2 produce fapte de nivel n + 1, ca și cînd faptele de nivel n au fost deja stabilite. Aceasta înseamnă că STINA-2 funcționează după strategia de dezvoltarea a căutării "în lățime".

De altfel, se observă din figura 3.17. că noile concluzii ale regulilor compatibile cu o stare a bazei de fapte sînt adunate succesiv în subansamblul numit FAPTENOI (instrucțiunea 4.3.). Dacă toate regulile au fost explorate, FAPTENOI este adăugată la BF (instrucțiunea 1.3). Pentru a încerca stabilirea faptului "Q", se lansează motorul STINA-2 prin apelul procedurii STAB-UN-FAPT ("Q"). Acțiunile motorului se desfășoară astfel:

- plecînd de la starea inițială a bazei de fapte {A, C, D, E, G, H, K}, declanșarea regulii 3 determină subansamblul FAPTENOI = {B};
- în continuare, declanșarea regulilor 6 și 9, folosind aceeași stare inițială a bazei de fapte, va determina un nou conținut pentru FAPTENOI = {B, R, F};
- întrucît nu mai rămîne nici o regulă declanșabilă pe starea inițială a bazei de fapte, FAPTENOI este adăugat la BF, care va deveni {A, C, D, E, G, H, K, B, R, F};
- în funcție de această nouă stare a bazei de fapte, se declanșează regula 4, care va stabili pe "Q".

Motorul STINA-2 reacționează în următoarea manieră în timpul ciclului inferențial:

- în faza de SELECTIE se elimină regula care a fost deja executată și se interzice utilizarea faptelor de nivel n + 1, atît timp cît faptele de nivel inferior sau egal cu n, nu s-au stabilit;
- fazele de FILTRARE și REZOLVARE A CONFLICTELOR sînt introduse prin procedura EXECUTA-UN-CICLU.

Motoarele STINA-1 și STINA-2 funcționează numai în regim irevocabil și monoton, întrucît declanșatorii din reguli nu sînt înlocuiți niciodată cu alții iar procedurile nu permit decît adăugarea de fapte (instrucțiunile 4.3).

3.3.4.2. Scheme de motoare bazate pe strategia de control înapoi, cu regim prin tentative și monotonie

Trebuie să menționăm că trebuie avute din nou în vedere convențiile precizate în paragraful anterior, cu privire la baza de cunoștințeși codul de scrierea programelor.

a) Motorul "STINAP-1", cu obținere de subprobleme prin strategia de dezvoltare a căutării "în-adîncime" are schema prezentată în fig.3.18.

Acest motor se lansează prin apelul uneia dintre procedurile STAB-UN-FAPT sau STAB-CONJUCT-FAPTE. De exemplu, pentru stabilirea faptului "Q" se apelează procedura STAB-UN-FAPT ("Q").

procedura STAB-UN-FAPT (FAPT)
1 dacă FAPT se află în BF, atunci restituie "succes"
2 restituie STAB1 (BR)

procedura STAB1 (REGULI, oregulă)
1 dacă REGULI este vidă, atunci restituie "eșec"
2 OREGULA ← alege o regulă oarecare din REGULI (de exemplu, prima întîlnită)
3 REGULI ← REGULI mai puțin OREGULA
4 dacă OREGULA conține FAPT în concluzie, atunci dacă STAB2 (OREGULA) = "succes", atunci restituie "succes"
5 restituie STAB1 (REGULI)

procedura STAB2 (REGULI, fapte)
1 FAPTE ← toate faptele din declanșatorul lui OREGULA
2 restituie STAB-CONJUCT-FAPTE (FAPTE)

procedura STAB-CONJUCT-FAPTE (FAPTE, unfapt)
1 dacă FAPTE este vidă, atunci restituie "succes"
2 UNFAPT ← alege un fapt oarecare din FAPTE (de exemplu, primul întîlnit)
3 FAPTE ← FAPTE mau puțin UNFAPT
4 dacă STAB-UN-FAPT (UNFAPT) = "eșec", atunci restituie "eșec"
5 restituie STAB-CONJUCT-FAPTE (FAPTE)

Fig. 3.18. Schema motorului STINAP-1, invocă reguli "în-înapoi"și obține subprobleme prin căutare "în-adîncime"

Motorul STINAP-1 invocă regulile prin înlănțuire "înapoi", el compară partea de concluzie a regulilor cu faptele de stabilit la un moment dat. Pentru acest tip de motor, partea de concluzie a regulilor (membrul drept) constituie declanșatorii. El memorează de fiecare dată un ansamblu de fapte de stabilit (o instanță) numit FAPTE. La apelul inițial al procedurii STAB-UN-FAPT ("Q"), ansamblul faptelor de stabilit nu va conține decît pe "Q". În aceste condiții, dacă regula 2 este invocată și declanșată, atunci faptul de stabilit "Q" va fi înlocuit cu faptele de stabilit I, J, L. Aceste fapte sînt numite subprobleme ale problemei "Q" și se spune că problema "Q" este problema tată pentru subproblemele I, J, L. În vederea stabilirii faptului "Q", motorul va obține și va aborda subproblemele în conformitate cu strategia de dezvoltare a căutării "în-adîncime", încercînd să rezolve cu prioritate subproblemele de pe nivelul cel mai înalt.

La acest tip de motoare, nu trebuie confundată declanșarea unei reguli cu tragerea concluziei unei reguli, întrucît, atunci cînd o regulă "în-înapoi" se folosește pentru stabilirea faptului care se află în partea sa de concluzie, plecînd de la faptele prezente în premisa deja dovedită, se spune că "se trage concluzia regulii", iar cînd o regulă "în-înapoi" se utilizează pentru înlocuirea faptelor prezente în partea sa de premisă, cu cele prezente în concluzie, se spune că "se declanșează regula".

Este interesantă de urmărit funcționarea acestui motor, STINAP-1, în condițiile stabilite în fig.3.12., în încercarea de stabilire a aceluiași fapt "Q".

Regula 2 din baza de reguli este declanșată. Faptul de stabilit, primul considerat "I", va determina declanșarea regulei 1, care va introduce faptele K, L, M. Deoarece "K" este deja stabilit și nici-o regulă nu mai conține "L" sau "M" în concluzia sa (cu rol de declanșator), faptul de stabilit redevine "I".

Întrucît nici o altă regulă nu are pe "I" în declanșator, faptul de stabilit redevine "Q". Se declanșează regula 4, din care se introduc faptele A și B. Deoarece A și B sînt deja stabilite, motorul va considera

faptul de stabilit "B", care determină concluzia (declanșatorul) regulii 3 să introducă faptele C, D, E. Aceste fapte sînt deja stabilite și regula 3 va conchide pe "B", apoi pe "Q" ca fiind stabilite. În fig.3.19. prezentăm graful SI-SAU pentru subproblemele care au stabilit pe "Q".

Strategia specifică motorului face parte din familia de strategii numită "dezvoltarea căutării în-adîncime" și constă din căutarea pe o ramură a arborescenței, de exemplu Q-a-I-b... pînă cînd se întîlnește un vîrf "K"; deoarece K este un fapt deja stabilit, se va comuta căutarea pe cea mai apropiată ramură, de exemplu b (ramificație de tip SI). În acest moment este posibilă o căutare de la rădăcină (=) spre L. Deoarece L nu este stabilit, se coboară la cea mai apropiată ramificație capabilă să permită declanșarea altei reguli, de exemplu I (de tip SAU); în acest punct nu există mai mult de o regulă care conchide pe I și se coboară căutarea la cele mai apropiate ramificații SAU (pe Q), punct în care se continuă căutarea pe ramura Q-c-A... .

Prezentăm în continuare ciclul de bază al acestui motor. În strategia de căutare specifică acestui motor, STINAP-1, declanșarea unei reguli constă în înlocuirea unui fapt stabilit (ipoteză, problemă, scop, obiectiv) cu alte fapte de stabilit, numite subprobleme. Subproblemele nu sînt înscrise explicit în baza de fapte, ci doar memorate prin intermediul apelurilor de proceduri. Subproblemele amînate momentan constituie o parte implicită a bazei de fapte, care va juca rolul determinant în alegerea regulii de executat în ciclul următor, cel mai apropiat.

În fig 3.20 prezentăm graful de stări obținut ca urmare a funcționării motorului STINAP-1 pentru stabilirea faptului "Q".

Urmărind stările bazei de fapte din acest graf, în partea dreaptă a fiecărui caz stare, observăm faptele stabilite (partea explicită a bazei de fapte) iar în partea stîngă observăm faptele de stabilit (partea implicită a bazei de fapte) sau subproblemele. Concluziile trase prin aplicarea regulilor 3 și apoi 2 nu sînt reprezentate în acest graf. Faza de SELECȚIE se materializează printr-o distincție între faptele stabilite (partea explicită) și subprobleme (partea implicită). Numai subproblemele vor fi comparate cu declanșatorii regulilor, în vederea declanșării.

De fapt, SELECȚIA indică printr-un pointer, la începutul fiecărui ciclu de bază, problema care trebuie să fie examinată; această problemă va fi luată în seamă dintre toate celelalte apărute în ciclul precedent (instrucți-unea 2 din procedura STAB-CONJUCT-FAPTE, din fig.3.18.). Pot exista excepții de la acest principiu, și anume:

- toate problemele obținute din ciclul precedent au fost recunoscute ca stabilite (restituire cu "succes" în instrucțiunile 1 - 5 ale procedurii STAB-CONJUNCT-FAPTE), caz în care problema care a determinat

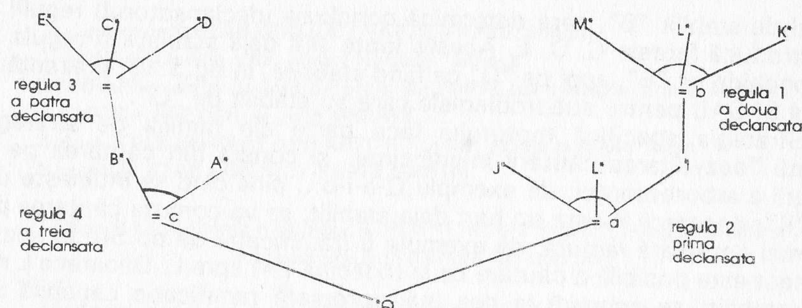


Fig.nr.3.19. Graful SI-SAU pentru subproblemele care stabilesc pe "Q", dezvoltat de către motorul STINAP-1

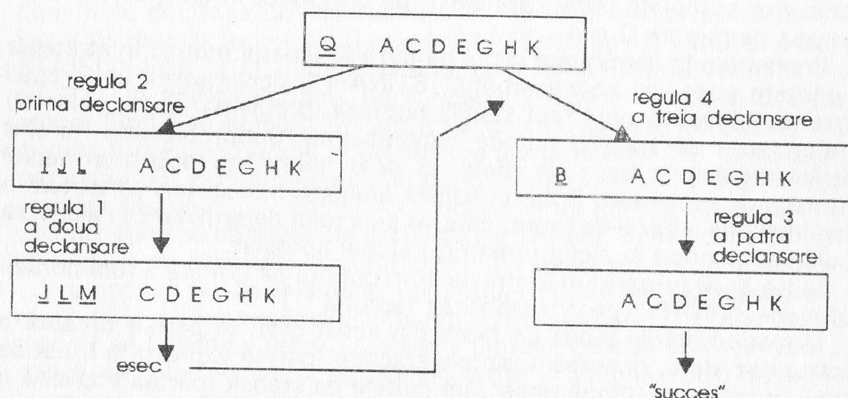


Fig. nr. 3.20. Graf de stări dezvoltate de motorul STINAP-1

declanșarea unei reguli în ciclul precedent este validată ca stabilită, prin restituirea de "succes" la apelul procedurii STAB-UN-FAPT;

- apariția unei probleme din ciclul precedent, care n-a putut fi stabilită (abandonarea procedurii STAB-CONJUNCT-FAPTE prin execuția instrucțiunii 4), caz în care problema care a determinat declanșarea ciclului precedent rămîne de stabilit (restituirea de "eșec" a procedurii STAB-UN-FAPT).

În ambele cazuri, SELECȚIA va propune o problemă rezultată din ciclul imediat anterior care va fi urmat. Motorul se oprește atunci cînd toate problemele inițiale s-au validat ca fiind "stabilite" sau cînd s-a

recunoscut că o problemă ori alta n-au putut fi stabilite. Există posibilitatea ca acest motor să rămînă în ciclu de funcționare perpetuă (în buclă) dacă adăugăm și regula a 10-a în baza de reguli inițială, astfel: Q ---> L.

Fazele de FILTRARE și REZOLVARE A CONFLICTELOR sînt introduse prin instrucțiunile 1 - 4 ale procedurii STAB1 iar etapa de EXECUȚIE este reprezentată de procedura STAB2.

b) Variante de îmbunătățire a motorului STINAP-1.

Prezentăm în continuare cîteva îmbunătățiri posibile ale acestui motor, deoarece sînt practicate intens la o mare majoritate a sistemelor expert din domeniul economic, inclusiv din contabilitate.

b1) Apelarea prin stabilirea unei disjuncții de fapte presupune introducerea procedurii STAB-DISJUNCT-FAPTE, astfel:

procedura STAB-DISJUNCT-FAPTE (FAPTE, unfapt)
1 dacă FAPTE este vidă, atunci restituie "eșec"
2 UNFAPT <--- primul din lista de FAPTE
3 FAPTE <--- FAPTE mai puțin UNFAPT
4 dacă STAB-UN-FAPT (UNFAPT) = "succes", atunci restituie "succes"
5 restituie STAB-DISJUNCT-FAPT (FAPTE)

Cu această procedură, STAB-DISJUNCT-FAPTE, motorul STINAP-1 poate fi lansat pentru stabilirea unei conjuncții de fapte ori pentru stabilirea unui fapt. Procedura va căuta să stabilească o disjuncție de fapte, în sensul că reușește și atunci cînd un fapt de stabilit se află printre celelalte din ansamblu.

b2) Memorarea faptelor deja stabilite și editarea de mesaje adecvate.

Pe măsura stabilirii faptelor, acestea se vor memora (ca să nu se mai încerce stabilirea lor) și se vor edita mesaje adecvate. Instrucțiunea 4 din procedura STAB1 (fig.3.18.) se va înlocui cu secvența:

4' dacă OREGULA conține pe FAPT în concluzie, atunci
4'.1 dacă STAB2 (OREGULA) = "succes", atunci
4'.2 început
4'.3 scrie că OREGULA a permis stabilirea lui FAPT
4'.4 adaugă FAPT in BF
4'.5 restituie "succes"
4'.6 sfîrșit

Evitarea intrării în buclă (ciclu infinit) a motorului se poate realiza prin interzicerea declanșării unei reguli de două ori și adăugarea în baza de fapte a faptelor deja stabilite. Putem face acest lucru în motorul STINAP-

1 prin inserarea între instrucțiunile 4'.3 și 4'.4 din procedura STAB1 îmbunătățită anterior a instrucțiunii 4'.3.1. astfel:

4'.3.1 BR ← BR mai puțin OREGULA

b3) Stabilirea unei cooperări între modulul explicativ și utilizator în timpul procesului de căutare.

În acest sens există trei posibilități și anume:

i) Prima posibilitate constă în dotarea motorului cu un comportament alternativ, în sensul că înainte de a abandona un fapt, să interogheze utilizatorul despre validitatea faptului.

Putem realiza aceasta prin înlocuirea instrucțiunii 1 din procedura STAB1, despre care am vorbit anterior, cu secvența de instrucțiuni:

1' dacă REGULI este vidă, atunci
1'.1 început
1'.2 cere utilizatorului să răspundă dacă poate afirma FAPT
1'.3 dacă răspunsul este negativ, atunci restituie "eșec"
1'.4 adaugă FAPT în BF
1'.5 restituie "succes"
1'.6 sfîrșit

Această secvență este posibilă numai în condițiile în care un fapt X este deductibil, adică există o regulă care conține faptul X în concluzie și fiecare din faptele premisei se află în baza de fapte sau este la rîndul său deductibil. Dacă faptul de stabilit X, rezultat al declanșării regulii R, lipsește din baza de fapte și este non-deductibil, atunci motorul abandonează problema X și caută să stabilească din nou concluzia regulii R, examinînd alte reguli de felul lui R.

ii) A doua posibilitate constă în interzicerea adresării de mai multe ori a aceleași întrebări utilizatorului.

În acest scop, se va memora într-un vector identificator numit BAZACERERI ansamblul faptelor la care s-au dat deja răspunsuri negative (cele cu răspunsuri pozitive s-au adăugat în baza de fapte), evitîndu-se deducția unui fapt deja introdus în acest vector. Cînd se lansează modulul explicativ, se va șterge acest vector concomitent cu schimbarea identificatorilor BF și BR. Această îmbunătățire presupune înlocuirea instrucțiunii "1" din procedura STAB1 cu următoarea secvență de instrucțiuni:

1" dacă REGULI este vidă, atunci
1".1 început

1".2 cere utilizatorului să răspundă dacă poate afirma FAPT
1".3 dacă răspunsul este negativ, atunci
1".4 început
1".5 adaugă FAPT în BAZACERERI
1".6 restituie "eșec"
1".7 sfîrșit
1".8 adaugă FAPT în BF
1".9 resituie "succes"
1".10 sfîrșit

De asemenea, între instrucțiunile 1 și 2 ale procedurii STAB-UN-FAPT (din fig.3.18.) se va introduce instrucțiunea:

1.1 dacă FAPT aparține lui BAZACERERI, atunci restituie "eșec"

iii) A treia posibilitate constă în exploatarea mai avantajoasă a vectorului BAZACERERI.

Pentru exploatarea mai avantajoasă a informației din BAZACERERI, vectorul cu fapte la care s-au dat răspunsuri negative, observăm că este inutilă căutarea stabilirii unui fapt din conjuncția de fapte de stabilit, dacă unul dintre fapte se află deja în BAZACERERI. În această situație, se va introduce în procedura STAB-CONJUNCT-FAPTE, între instrucțiunile 1 și 2, următoarea instrucțiune:

1' dacă un fapt din FAPTE aparține lui BAZACERERI, atunci restituie "eșec"

Alt rafinament mai este încă posibil prin distincția între cazul în care utilizatorul nu știe să răspundă la o cerere și cazul în care va amîna răspunsul pentru mai tîrziu. Acest rafinament, în legătură cu interogarea utilizatorului, poate fi definit mai corect numai în condițiile unei aplicații concrete, prin așa-numitul "protocol de interogare adaptat aplicației". De pildă, în sistemul MYCIN, faptele sînt încadrate în una din categoriile:

- fapt exclusiv chestionabil și inutil de căutat reguli pentru deducția sa;
- fapt non-chestionabil, care se obține numai prin deducție;
- fapt chestionabil cu prioritate înaintea utilizării regulilor;
- fapt chestionabil numai în cazul eșecului regulilor care conchid asupra sa.

Ar fi deosebit de interesant de încercat dotarea motorului STINAP-1 cu protocolul următor: nu se pune prima întrebare utilizatorului, decît

după verificarea dacă faptul inițial căutat nu poate fi dedus prin alt mod. Se poate încerca și minimizarea numărului de întrebări, care pot fi puse în timpul căutării. În condițiile acestor idei, motorul STINAP-1 se poate programa și prin:

- înlocuirea instrucțiunii 2 din procedura STAB-UN-FAPT cu instrucțiunile:

2' WREGULI <--- toate regulile din BR care conțin FAPT în concluzie
3' restituie STAB1 (WREGULI)

- înlocuirea instrucțiunii 4 din procedura STAB1 cu instrucțiunea:

4' dacă STAB2 (OREGULA) = "succes", atunci restituie "succes"

În aceste condiții, instrucțiunea 2' de mai sus urmărește punerea în evidență a fazei de FILTRARE a ciclului de bază și a fazei REZOLVAREA CONFLICTELOR.

b) Motorul STINAP-2, dotat cu o modificare a strategiei de dezvoltare a căutării-în-adîncime permite căutarea în adîncime numai atunci cînd regula nu este imediat concludentă (vezi fig.3.21.).

procedura STAB-UN-FAPT (FAPT, dreguli)
1 dacă FAPT face parte din BF, atunci restituie "succes"
2 DREGULI ← toate regulile din BR care conțin FAPT în partea de concluzie
3 dacă DIRECTSTAB1 (REGULI) = "succes", atunci restituie "succes"
4 restituie STAB1 (REGULI)

procedura DIRECTSTAB1 (DREGULI, OREGULA)
1 dacă REGULI este vidă, atunci restituie "eșec"
2 OREGULA ← alege o regulă oarecare din REGULI (de exemplu, prima întîlnită)
3 REGULI ← REGULI mai puțin OREGULA
4 dacă DIRECTSTAB2 (OREGULA) = "succes", atunci restituie "succes"
5 restituie DIRECTSTAB1 (REGULI)

procedura DIRECTSTAB2 (OREGULA, dfapte)
1 DFAPTE ← toate faptele din premisa lui REGULA
2 restituie DIRECTSTAB-CONJUCT-FAPTE (DFAPTE)

procedura DIRECTSTAB-CONJUCT-FAPTE (FAPTE, unfapt)

1 dacă FAPTE este vidă, atunci restituie "succes"
2 UNFAPT ← alege un fapt oarecare din FAPTE (de exemplu, primul întîlnit)
3 FAPTE ← FAPTE mai puțin UNFAPT
4 dacă UNFAPT nu face parte din BF, atunci restituie "eșec"
5 restituie DIRECTSTAB-CONJUCT-FAPTE (FAPTE)

procedura STAB1 (REGULI, oregula)
1 dacă REGULI este vidă, atunci restituie "eșec"
2 OREGULA ← alege o regulă oarecare din REGULI (de exemplu, prima întîlnită)
3 REGULI ← REGULI mai puțin OREGULA
4 dacă STAB2 (OREGULA) = "succes", atunci restituie "succes"
5 restituie STAB1 (REGULI)

procedura STAB2 (LREGULA, dfapte)
1 DFAPTE ← toate faptele din premisa lui LREGULA
2 restituie STAB-CONJUCT-FAPTE (DFAPTE)

procedura STAB-CONJUNCT-FAPTE (FAPTE, unfapt)
1 dacă FAPTE este vidă, atunci restituie "succes"
2 UNFAPT ← alege un fapt oarecare din FAPTE (de exemplu, primul întîlnit)
3 FAPTE ← FAPTE mai puțin UNFAPT
4 dacă STAB-UN-FAPT (UNFAPT) = "eșec", atunci restituie "eșec"
5 restituie STAB-CONJUNCT-FAPTE (FAPTE)

*Fig. nr. 3.21.*Schema motorului STINAP-2. Produce subprobleme numai atunci cînd regula nu este imediat concludentă.

Urmărind cu atenție fig.3.21. observăm că, dispunînd de ansamblul regulilor care conchid un FAPT (instrucțiunea 2 din procedura STAB-UN-FAPT) se preferă mai întîi examinarea regulii și evidențierea dacă faptele din premisă există deja în baza de fapte BF (instrucțiunea 3 din aceeași procedură), adică dacă regula invocată este imediat concludentă.

Căutarea regulilor care conchid un FAPT se face prin deplasarea și examinarea mai întîi "în-lățime", pe un graf asemănător celui din fig. 3.22

Căutarea va avea loc numai după eșecul complet al căutării care a încercat să stabilească faptele care lipsesc din una din reguli. Această operație corespunde mai bine unei căutări la următorul nivel "în-adînci-

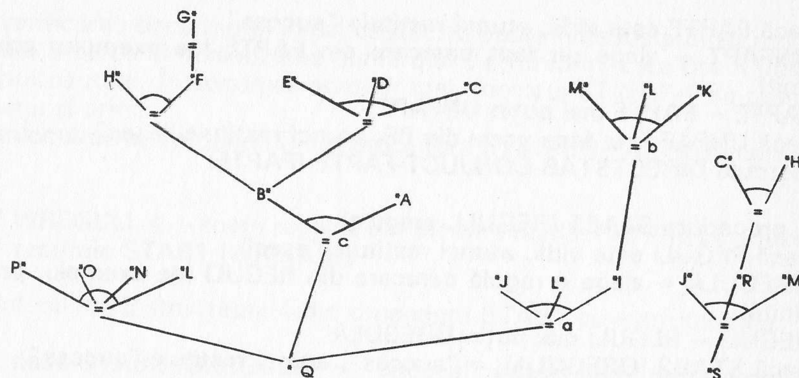


Fig. nr. 3.22. Graf SI-SAU dezvoltat de motorul STINAP-2

operație corespunde mai bine unei căutări la următorul nivel "în-adîncime", față de vîrful reprezentativ al FAPT-ului.

Motoarele STINAP-1 și STINAP-2 funcționează în regim prin tentative și monotonie. Nu se adaugă și nici nu se elimină fapte, iar faptele eventual adăugate nu introduc contradicții, în sensul că nu determină eliminări de fapte.

3.3.4.3. Schemă de motor de inferențe bazat pe strategia de control mixtă, cu regim prin tentative și non-monotonie (STMIXT)

Pentru demonstrarea acestui tip de motor ne trebuie o altă bază de cunoștințe, aceea din fig. 3.23, de mai jos.Baza de reguli (BR):

R1: P1 ← A1, A2
R2: P1 ← A3
R3: P2 ← P1, A4, P3
R4: P3, F1 ← A5, P4
R5: P3, F2 ← P5
R6: P3 ← A4, P6
R7: P4, F2, F3 ← A2

Baza de fapte stabilite (BF):
{F3, F4}

Baza de fapte de stabilit (BFS):
lista (P2)

Tabela acțiunilor terminale (probleme primitive)

Simbol	Fapte adăugate	Fapte scoase
A1	F1, F2	nimic
A2	nimic	F1
A3	F1	nimic
A4	nimic	nimic
A5	F2	F4

Fig. nr. 3.23. Baza inițială de cunoștințe pentru motorul de inferențe "STMIXT"

Baza de cunoștințe prezentată mai sus reprezintă simbolic o expertiză care poate fi utilizată la generarea planurilor de acțiuni cu ajutorul unui motor de inferențe pe care îl numim "STMIXT" și a cărui schemă o prezentăm în fig.3.24.

Dar, să vedem întîi unele precizări necesare cu privire la baza de reguli și caracteristicile motorului.

Regula R1 se interpretează astfel:

"pentru a rezolva problema P1 este suficient să se execute acțiunea A1 și apoi acțiunea A2"

Regula R4 se interpretează astfel:

"pentru a rezolva problema P3, știind că F1 este stabilit, este suficient să se execute acțiunea A5 și apoi să se rezolve problema P4."

Examinînd membrul stîng al regulilor (declanșatorul), observăm că fiecare element este un "filtru" reprezentat printr-un simbol. De asemenea constatăm că primul filtru se referă la o problemă de rezolvat întrucît face parte din baza de fapte de stabilit (BFS) sau lista problemelor de rezolvat (prescurtat de noi PDR). Următoarele filtre se referă la fapte stabilite din BF.

Regulile sînt invocate după strategia de control mixtă (combinată), în sensul că "filtrele" declanșatorului pot conține simultan fapte stabilite și fapte de stabilit. În absența filtrelor relative la faptele stabilite (BF), motorul va funcționa după strategia de control înapoi. Examinînd membrul drept al regulilor, "corpul", observăm că este definit prin acțiunile de realizat și problemele de rezolvat din BFS sau PDR. De exemplu, dacă primul element din PDR (BFS) este P1, regula R1 poate fi declanșată iar

A1 şi A2 se vor plasa în prima şi respectiv a doua poziţie, în timp ce P1 va fi retrasă din PDR. A1, A2..., A5 sînt considerate ca probleme imediat rezolvabile sau "probleme primitive", adică acţiuni elementare dintr-un "plan de acţiuni" sau "acţiuni terminale". Dacă o asemenea acţiune se trece într-un plan în curs de elaborare, se vor reprezenta şi efectele sale, în sensul adăugării şi scoaterii de fapte. În exemplul nostru, adăugarea înseamnă "stabilirea unui fapt" iar scoaterea înseamnă că "faptul nu este sau nu este încă stabilit".

O acţiune terminal este recunoscută dacă este prezentă în lista acţiunilor terminale (vezi fig.3.23.). De exemplu regula R4 înseamnă adăugarea lui F2, scoaterea lui F4 şi rezolvarea problemei P4, întrucît A5 care trebuie executată se află în tabela acţiunilor terminale şi implică asemenea operaţii. Cînd, după declanşarea unei reguli, o acţiune terminal se află pe prima poziţie sau în "vîrful" lui PDR (BFS), ea va fi scoasă şi se va aşeza la sfîrşitul listei numită PLAN, şi imediat are loc o actualizare a descrierilor din tabela acţiunilor terminale. De exemplu, cînd A5 este scoasă, se adaugă F2 în BF şi se scoate F4. În această manieră se realizează planificarea sau "generarea planurilor", care nu-i lipsită de interes în previziunile contabile şi chiar în expertiza contabilă.

Motorul STMIXT generează planuri de acţiuni, deoarece dispune de posibilitatea de a comanda prin redactarea unei reguli şi prin informaţiile asociate în tabela acţiunilor terminale, obţinerea unor concluzii stabilite anterior prin aplicarea altor reguli ori scoaterea din PLAN a unei acţiuni terminal introdusă anterior în planul în curs de elaborare. În acest caz, mecanismul de control înapoi a motorului provoacă reconsiderarea adăugărilor şi scoaterilor de fapte, eventual asociate la o acţiune terminal. În acelaşi timp, dacă regulile nu comandă decît adăugări, reconsiderarea acestor reguli se va materializa prin scoaterea faptelor stabilite.

Cînd un motor de inferenţe reconsideră faptele stabilite, se spune că este "non-monoton". Motorul STMIXT, din exemplul nostru, este "non-monoton" şi se poate lansa prin apelul procedurii REZOLV-O-PROBL (P2), dacă avem în vedere schema din fig.3.24.

procedura REZOLV-O-PROBL (PROBLEMA)
1 PDR ← lista cu un singur element în PROBLEMA
2 PLAN ← lista vidă
3 UNCONTEXT ← lista cu elementele PDR, BF, PLAN
4 dacă REZOLV (UNCONTEXT) = "eşec", atunci restituie "eşec"
5 editează PLAN
6 restituie "succes"

procedura REZOLV (CONTEXT, INDIC-REGULI, uncontext)
1 dacă PDR este vidă, atunci restituie "succes"
2 INDIC-REGULI ← EXEC-UN-CICLU (început PDR, INDIC-REGULI)
3 dacă INDIC-REGULI = "eşec", atunci restituie "eşec"
4 UNCONTEXT ← lista cu elementele PDR, BF, PLAN, care există la reîntoarcerea din EXEC-UN-CICLU
5 dacă REZOLV (UNCONTEXT, BR) = "succes", atunci restituie "succes"
6 dacă INDIC-REGULI = "primitivă", atunci restituie "eşec"
7 PDR ← primul element din CONTEXT
8 BF ← al doilea element din CONTEXT
9 PLAN ← al treilea element din CONTEXT
10 INDIC-REGULI ← INDIC-REGULI mai puţin primul element
11 restituie REZOLV (CONTEXT, INDIC-REGULI)

procedura EXEC-UN-CICLU (PROBLEMA, REGULI, oregula)
1 dacă PROBLEMA este prezentă în tabela acţiunilor terminale, atunci
1.1 început
1.2 suprimă începutul lui PDR (adică PROBLEMA)
1.3 aşează PROBLEMA pe ultimul loc în PLAN
1.4 execută adăugare şi scoatere din BF a faptelor descrise PROBLEMA în tabela acţiunilor terminale
1.5 restituie "primitivă"
1.6 sfîrşit
2 dacă REGULI este vidă, atunci restituie "eşec"
3 OREGULA ← prima regulă din REGULI
4 dacă OREGULA este declanşabilă (are declanşator compatibil cu PROBLEMA şi BF), atunci
4.1 început
4.2 suprimă începutul lui PDR
4.3 introduce elementele corpului lui OREGULA în PDR (primul la început şi apoi următoarele)
4.4 restituie REGULI
4.5 sfîrşit
5 restituie EXEC-UN-CICLU (PROBLEMA, REGULI mai puţin OREGULA)

Fig. nr. 3.24. Schema motorului "STMIXT", care invocă regulile conform strategiei de control mixte, cu regim prin tentative şi non-monotonie

Motorul prezentat în acest exemplu declanşează prima regulă din BR, al cărei declanşator este compatibil cu PDR şi BF. Regula declanşată în

continuare va fi prima din BR compatibilă cu situația creată ca urmare a declanșării precedentei; se spune că are loc o declanșare prin căutare "întîi în-adîncime". Dacă PDR viitoare va fi vidă, motorul se oprește deoarece problema inițială a fost rezolvată iar starea curentă a PLAN-ului va fi și cea finală. Dacă PDR nu este vidă, și nici o altă regulă nu mai este declanșabilă, motorul va încerca să execute o "întoarcere-înapoi" și să aplice o nouă regulă asupra ultimei probleme non-primitive scoasă din PDR. În felul acesta, STMIXT va restabili cu exactitate contextul (PDR, BF, PLAN), care exista în momentul cînd problema a fost atacată, înainte de a începe o nouă problemă și dacă este posibil printr-o altă regulă de descompunere.

Funcționarea motorului STMIXT rezultă din observarea grafului SI-SAU prezentat în fig.3.25. care urmează.

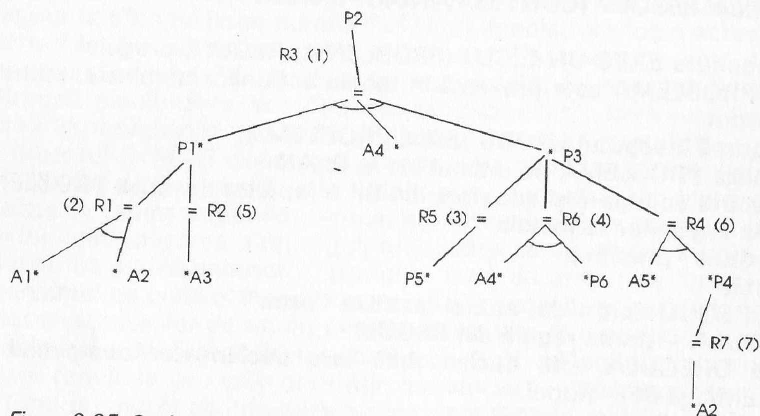


Fig.nr. 3.25. Graful SI-SAU dezvoltat de către STMIXT. Fiecare regulă are în paranteză rangul de invocare

În conformitate cu acest graf (fig.3.25.), efectele declanșării regulilor se prezintă astfel:

1) După declanșarea regulii R3:
PDR = (P1, A4, P3), BF rămîne neschimbată iar PLAN vidă.

2) După declanșarea regulii R1:
PDR = (A1, A2, A4, P3), BF este neschimbată iar PLAN vidă.
Se vor scoate A1, A2 și A4 cu efectele asociate asupra lui BF, fapt care determină PDR = (P3), BF = {F2, F3, F4} și PLAN = (A1, A2, A4).

3) După declanșarea regulii R5:
PDR = (P5), BF = neschimbată, PLAN = neschimbat. Nici o regulă

nu este declanșabilă în acest context. Motorul revine la rezolvarea problemei (P3), de unde:
PDR = (P3), BF = {F2, F3, F4} și PLAN = (A1, A2, A4).

4) după declanșarea regulei R6:
PDR = (A4, P6), BF = neschimbat și PLAN = neschimbat, va avea loc scoaterea lui A4, cu efectele asociate asupra lui BF, PDR = (P6), BF = neschimbat și PLAN = (A1, A2, A4, A4). Nici-o regulă nu este declanșabilă în acest nou context iar motorul revine la rezolvarea problemei P3, de unde se obține:
PDR = (P3), BF = {F2, F3, F4}, PLAN = (A1, A2, A4).
În acest context nici-o regulă nu este declanșabilă și motorul revine la rezolvarea problemei P1, de unde se obține:
PDR = (P1, A4, P3), BF = {F3, F4} și PLAN = vid.

5) După declanșarea regulii R2:
PDR = (A3, A4, P3), BF = neschimbată și PLAN = vid; va avea loc scoaterea lui A3 și A4, cu efecte asociate asupra lui BF:
PDR = (P3), BF = {F1, F3, F4} și PLAN = (A3, A4).

6) După declanșarea regulii R4:
PDR = (A5, A4), BF = neschimbată, PLAN = neschimbat; va avea loc scoaterea lui A5, cu efectele asociate asupra lui BF:
PDR = (P4), BF = {F1, F2, F3} și PLAN = (A3, A4, A5).

7) După declanșarea regulii R7:
PDR = (A2), BF = neschimbată și PLAN = neschimbat; se va scoate A2, cu efectele asupra lui BF:
PDR = vidă și deci, motorul se oprește în contextul:
BF = {F1, F2, F3}, PLAN = (A3, A4, A5, A2) ceea ce ne-a interesat să rezolvăm.

Procedura REZOLV din "STMIXT" încearcă să reducă problema la începutul sau "vîrful" lui PDR, făcînd apel la procedura EXEC-UN-CICLU. În această situație, există patru cazuri care pot fi întîlnite:

1) Procedura EXEC-UN-CICLU restituie "eșec", dacă problema nu corespunde unei acțiuni terminale și nu se mai descompune prin nici-o regulă disponibilă din lista INDIC-REGULI. În acest caz, procedura REZOLV curentă se va opri, restituind "eșec" la nivelul superior, iar INDIC-REGULI va fi obținută ca al doilea argument de apel al procedurii REZOLV. Dacă procedura EXEC-UN-CICLU nu restituie "eșec", înseamnă că problema este o acțiune terminală sau o problemă decompozabilă, prin una din regulile listei obținute prin apelarea procedurii EXEC-UN-CICLU. În aceste situații se lansează procedura REZOLV (cu un ultim apel, pe care-l numim A, care are un nou context format din evoluția lui BF și/PDR și BR în întregime.

2) Dacă acest ultim apel A, al procedurii REZOLV, restituie "succes", procedura REZOLV curentă va restitui "succes" la nivelul superior. În caz contrar, prelucrarea diferă numai prin procedura EXEC-UN-CICLU, care recunoaște problema ca pe o acțiune terminală (problemă primitivă, nedecompozabilă) sau încearcă decompoziția.

3) Procedura EXEC-UN-CICLU va restitui "primitiva", dacă problema este o acțiune terminală. În acest caz, eșecul apelului A al procedurii REZOLV arată că trebuie considerată punerea problemei la începutul lui PDR, și se va restitui "eșec" la nivelul superior.

4) Dacă problema nu este o acțiune terminală, înseamnă că este decompozabilă. Procedura EXEC-UN-CICLU introduce rezultatul decompoziției la începutul listei PDR și restituie sublista bazei de reguli, care începe prin regula declanșată. În acest caz, eșecul apelului A, al procedurii REZOLV, poate fi determinat de introducerea subproblemelor problemei decompozabile. Ca urmare, este necesară găsirea altei descompuneri; contextul dinaintea apelului procedurii EXEC-UN-CICLU este restabilit și un nou apel A este lansat. Primul argument al acestui nou apel va fi contextul restaurat iar al doilea argument este lista regulilor din procedura REZOLV curentă, mai puțin regula deja încercată.

Motorul de inferențe "STMIXT" poate fi îmbunătățit în așa fel încît să poată face o ordonare a problemelor pe care le rezolvă, să funcționeze interactiv, să controleze dezvoltarea căutării în adîncime, să editeze toate soluțiile de planuri posibile etc.

Pentru ordonarea problemelor este suficientă revizuirea procedurii REZOLV-O-PROBL astfel încît PDR să fie încărcată inițial cu problema considerată necesară.

Funcționare interactivă se poate realiza dacă, înaintea unei întoarceri-înapoi se introduce interogarea utilizatorului, editînd începutul listei PDR și numai o parte sau toată BF, astfel încît utilizatorul să poată menționa că problema supusă atenției este rezolvabilă în condițiile precizate.

Pentru controlarea dezvoltării căutării "în-adîncime", astfel încît prin declanșarea de mai multe ori a unei reguli în același context, să nu se intre în ciclu infinit (buclaj), se poate asocia fiecărui element din PDR un "număr de nivel" egal cu nivelul problemei din al cărei descompunere s-a obținut problema, mărit cu 1. Astfel, o problemă inițială poate avea numărul de nivel 0, celelalte 1, 2 ș.a.m.d. Se va interzice atacarea unei probleme al cărui nivel depășește o valoare limită stabilită anterior. Nivelul unei probleme poate fi diminuat în cursul căutării.

În principiu, pot exista mai multe planuri de acțiuni susceptibile pentru reprezentarea soluției unei probleme date. De exemplu, conform bazei de cunoștințe din fig.3.23, referitoare la motorul "STMIXT", se pot

găsi două planuri pentru rezolvarea problemei P1, planul (A1, A2) și planul (A3). Dacă se adaugă și a opta regulă, să zicem P1 <--- A1, atunci cele două planuri pentru rezolvarea problemei P2 vor fi:

(A3, A4, A5, A2) și (A1, A4, A5, A2)

Motorul "STMIXT" se poate modifica astfel încît, la cerere, să caute succesiv aceste două planuri. Pentru aceasta, se suprimă instrucțiunea 5 din procedura REZOLV-O-PROBL și se înlocuiește cu instrucțiunea 1 din procedura REZOLV, astfel:

1 dacă PDR este vidă, atunci
1.1 început
1.2 editează PLAN
1.3 cere utilizatorului dacă dorește alt plan
1.4 dacă răspunsul este afirmativ, atunci restituie "eșec"
1.5 restituie "eșec"
1.6 sfîrșit

Se observă că, instrucțiunea 1.4 forțează un "eșec", fapt care obligă motorul să caute o altă soluție eventuală, plecînd de la contextul curent.

Sînt importante de analizat și alte insuficiențe ale motorului STMIXT, cum ar fi:
- rezolvarea problemelor de identificare sau de diagnostic;
- rezolvarea problemelor neordonate;
- rezolvarea problemelor "autopuse" și a celor interdependente; toate foarte importante și în contabilitate.

3.4. Dezvoltarea sistemelor expert

Dezvoltarea sistemelor expert cunoaște în prezent o mare vogă în cercul informaticienilor, care percep aceste sisteme ca pe niște mijloace concrete de informatizare a activităților intelectual creative din toate domeniile.

Pentru domeniul contabilității, al gestiunii în general, fără nici-o îndoială, dezvoltarea sistemelor expert ridică aceleași probleme de inginerie a cunoașterii, de achiziție și reprezentare a cunoștințelor, de respectare a fazelor de dezvoltare, a stadiilor de dezvoltare și ocolirea unor eventuale primejdii, cum sînt alegerea unor probleme necorespunzătoare, lipsa tehnicii de calcul adecvate, a resurselor financiare, a expertului în domeniu, formularea unor cereri excesive ș.a.

Dezvoltarea sistemelor expert face obiectul celor mai intense cercetări în multe organizații, tocmai datorită interesului deosebit pentru această tehnologie de vîrf.

În prezent, există puţine metodologii pentru dezvoltarea sistemelor expert. Cele mai cunoscute sînt:

- GURU, realizată în SUA la firma Micro Data Base Systems;
- ES-SDEM, Software Development Engineering Methodology for Expert Systems, realizată în Japonia, în cadrul companiei FUJITSU;
- IBM-ESE, un mediu de dezvoltare a sistemelor expert, care dovedeşte existenţa unei metodologii bine disimulată, pentru menţinerea în fruntea competiţiei;
- DEC, Digital Equipment Corporation, companie transnaţională cu sediul în SUA, care posedă o metodologie proprie, de mare eficienţă, din moment ce a elaborat cu începere din 1980 peste 30 de sisteme expert pentru folos propriu.

În literatura de specialitate nu există decît scurte rezumate ale unor asemenea metodologii sau doar ghiduri de utilizare însoţite de suporţi magnetici care conţin limbaje shell. În publicaţiile companiei FUJITSU se arată că, metodologia constituie proprietatea firmei şi este deci, o documentaţie de lucru internă. În conformitate cu această metodologie[1], procesul de dezvoltare a sistemelor expert se bazează pe metodologia de proiectare a sistemelor informatice convenţionale, în cadrul căreia etapele de proiectare de detaliu şi elaborarea programelor s-au înlocuit cu etapa de proiectare a prototipului.

Metodologia GURU, utilizată pe scară largă în mai multe ţări din lume, nu este nici ea prezentată detaliat. Cartea celor doi oameni de ştiinţă americani C.Holsapple[2] şi A.Whinston, specialişti în informatica economică şi management, prezintă doar un ghid pentru manageri din perspectiva dezvoltării sistemelor expert, ca suport pentru decizii. Se oferă cititorilor conceptele şi tehnicile inteligenţei artificiale în contextul rezolvării problemelor economice.

Donald A. Waterman a editat de asemenea un ghid[3] în sistemele expert, în cadrul căruia oferă unele detalii privind dezvoltarea.

Vom prezenta şi noi în continuare o sinteză din ceea ce numeşte acest autor dezvoltarea (construirea) sistemelor expert.

---

[1] Matsumoto, S., "ES/SDEM - Software Development Engineering Methodology for Expert Systems", in Future Generation Computer Systems, 5(1989), North-Holland, p. 33-39.

[2] Holsapple, C.H., Whinston, A.B., "Manager's Guide to Expert Systems using GURU", Dow Jones Irwin, 1987.

[3] Waterman, D.A., "A Guide to Expert Systems", Addison Wesley Pub.Co., Reading, Massachusetts, 1986.

3.4.1. Criterii şi stadii de dezvoltare a sistemelor expert

Criteriile de dezvoltare a sistemelor expert sînt: posibilitatea, justificabilitatea şi oportunitatea.

a) Criteriul posibilităţii se referă la faptul dacă un sistem expert este capabil să rezolve o problemă concretă dintr-un anumit domeniu. În principiu, posibilitatea dezvoltării unui sistem expert, pentru o anumită clasă de probleme, derivă din conjuncţia următoarelor aserţiuni, care trebuie verificate în prealabil:

- problema cere pricepere cognitivă;
- problema nu este dintre cele rutiniere;
- problema necesită expertiză, după unele metode specifice;
- există experţi adevăraţi pentru rezolvarea unor asemenea probleme;
- experţii sînt de acord cu folosirea sistemelor expert;
- sarcinile specifice problemei nu sînt extrem de dificile şi pot fi bine înţelese;

Cercetătorii în inteligenţa artificială sînt de acord cu faptul că, în prealabil, trebuie realizată o descriere a specificului problemei, observîndu-se în principal potenţialul, scopul şi laturile proprii dezvoltării sistemului expert. Cu acest prilej, expertul trebuie să poată emite şi explica metodele cu care soluţionează problema.

b) Criteriul justificabilităţii se referă la faptul dacă dezvoltarea unui sistem expert se justifică pentru o anumită problemă sau clasă de probleme. Justificabilitatea derivă din următoarele aserţiuni disjuncte:

- problema de soluţionat are o înaltă rentabilitate;
- expertiza umană poate fi pierdută datorită schimbării personalului;
- există puţini experţi în domeniu;
- expertiza este necesară în mai multe locuri concomitent, din cadrul întreprinderii;
- expertiza este necesară în mediu ostil (toxic, etc.).

c) Criteriul oportunităţii dezvoltării sistemului expert se referă la următorii factori cheie: natura, complexitatea şi scopul problemei de rezolvat, care acţionează în conjuncţie.

"Natura" se referă la faptul dacă problema necesită prelucrări simbolice şi respectiv soluţii euristice. Complexitatea vizează gradul de dificultate iar scopul are în vedere valoarea practică şi dimensiunile problemei.

Problemele rezolvabile pe cale algoritmică, prin proceduri informatice clasice, care garantează soluţii rapide şi exacte în fiecare moment, nu sînt bune candidate pentru dezvoltarea de sisteme expert. Dacă totuşi, metodele de soluţionare sînt de mare complexitate şi dificultate, atunci

se pot folosi sisteme expert.

Complexitatea există atunci cînd pentru anumite probleme, omul are nevoie de ani de muncă pentru a căpăta statut de expert, iar cheltuielile cu pregătirea sa ar fi prohibitiv de mari. De asemenea alegerea corectă a scopului este crucială pentru reușita dezvoltării sistemului expert.

Numai dacă o problemă a îndeplinit toate aceste criterii de dezvoltare a sistemelor expert se vor începe fazele de dezvoltare, printr-o abordare incrementală.

Abordarea incrementală a dezvoltării sistemelor expert, înseamnă trecerea succesivă de la un stadiu evolutiv la altul, în care sistemul însuși asistă procesul de dezvoltare, operînd schimbările necesare. Se începe cu un "prototip demonstrativ", care este un mic program de inteligență artificială demonstrativ, pentru soluționarea unei părți din problemă. Prototipul se utilizează în două scopuri:

a)pentru a convinge utilizatorii de potențialul oferit de tehnologia sistemelor expert, inclusiv în problema concretă;

b) pentru tratarea conceptelor despre problemă, scopul și reprezentarea cunoașterii specifice problemei.

Sistemele expert prototip demonstrativ pot avea circa 50-100 de reguli, care se vor executa pe una sau pe două cazuri specifice. Prototipul demonstrativ necesită circa trei luni pentru dezvoltarea sa.

Următorul stadiu este acela de "prototip de cercetare", concretizat într-un sistem expert de dimensiune medie, capabil de performanțe credibile, asupra căruia se aplică o testare extinsă, incluzîndu-se noi performanțe pentru satisfacerea cît mai bună a scopului general. Acest prototip poate avea între 200 și 500 de reguli și necesită circa 1-2 ani de muncă intensă.

Urmează stadiul de "prototip în domeniu", care poate avea circa 500-1000 de reguli și necesită 2-3 ani de muncă. Sistemului expert din acest stadiu i se observă comportamentul în domeniu, în toate detaliile de utilizare posibile, notîndu-se dificultățile care pot apare și luîndu-se măsurile de corectare adecvate. De regulă, sistemele expert în acest stadiu funcționează lent și neeficient.

Stadiul următor este acela de "prototip de producție", sistem expert de mari dimensiuni, care se listează în extenso, se testează și se reimplementează în cele mai eficiente limbaje, în scopul creșterii vitezei de lucru și reducerii spațiului de memorie ocupată. În acest stadiu, sistemul expert funcționează cu o acuratețe bună și ia decizii eficiente. Va conține între 500 și 1500 de reguli, iar durata de dezvoltare va ajunge la circa 4 ani.

Cele mai puține sisteme expert pot ajunge în stadiul de "sisteme

comerciale", destinate vînzării către beneficiarii interesați. Circa 6 ani durează dezvoltarea unui sistem expert comercial, cu posibilități garantate în exploatare, deoarece își execută eficient toate funcțiile, cu viteză optimă, preia în mod elegant cererile utilizatorilor și conduce dialogul om-mașină pentru rezolvarea problemei specifice.

3.4.2.Faze de dezvoltare a sistemelor expert

Dezvoltarea sistemelor expert are loc în cinci faze interdependente și suprapuse,după cum urmează: identificarea, conceptualizarea, formalizarea, implementarea și testarea. În fig.3.26. prezentăm aceste faze.

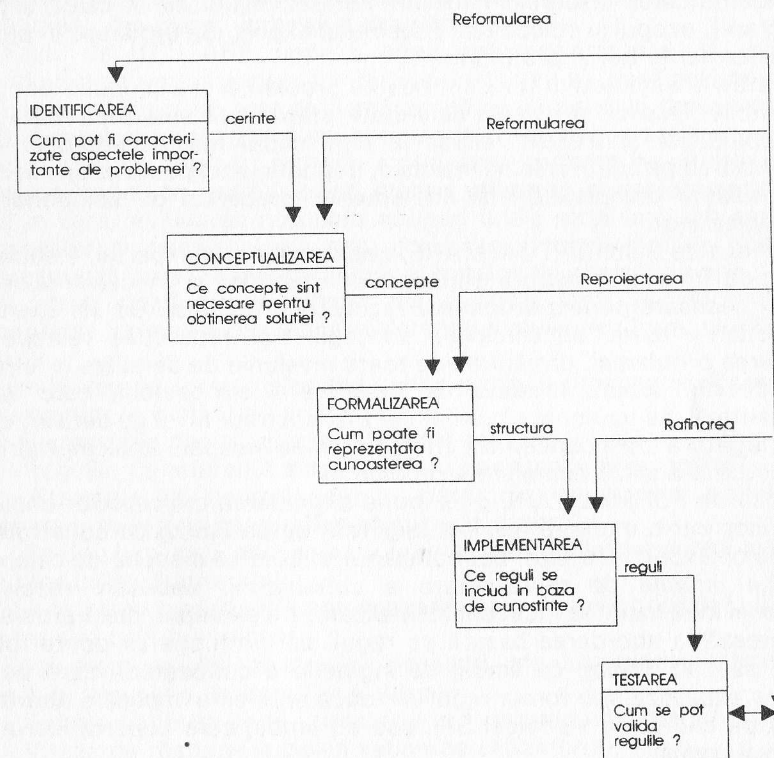


Fig.nr.3.26. Fazele dezvoltării sistemelor expert.

Distincția dintre aceste faze prezintă interes didactic întrucît au loc reveniri permanente la fazele precedente și nu-i ușor de indicat ordinea în care ele trebuie să se desfășoare. Oricînd, în timpul dezvoltării sistemului, inginerul de cunoștințe (cognoticianul) poate fi angajat în oricare dintre faze.

În faza de IDENTIFICARE, cognoticianul și expertul lucrează împreună, pentru determinarea tuturor aspectelor mai importante ale problemei:

- delimitarea corectă a problemei însăși (tipul și scopul);
- identificarea participanților la procesul de dezvoltare a sistemului expert, eventual și altor experți suplimentari;
- identificarea resurselor necesare (timpul, mijloacele de calcul și teleprelucrare), scopul și obiectivele sistemului expert. Se urmăresc în principal performanțe noi și profit sporit.

Dintre aceste activități, identificarea problemei și a scopului sînt cele mai dificile. Deseori problema delimitată prima dată este prea mare sau prea complexă și trebuie redusă la dimensiuni mai acceptabile. Prin concentrarea pe subprobleme mai mici, cognoticianul poate obține în mod rapid măsura complexității și implementa proceduri de soluționare a subproblemelor.

În faza de CONCEPTUALIZARE, cognoticianul și expertul, trebuie să se decidă împreună asupra conceptelor, relațiilor și mecanismelor de control necesare pentru descrierea rezolvării problemei. Se vor explora, cu răbdare, toate subsarcinile, strategiile și restricțiile relative la rezolvarea problemei, urmărindu-se toate nivelurile de detaliere și luîndu-se o decizie asupra nivelului de detaliere a cunoașterii, care va fi reprezentată. Se va scoate în relief cel mai abstract nivel de detaliu, care indică legătura între conceptele cheie. Înaintea începerii implementării se va efectua o analiză completă a problemei.

Faza de FORMALIZARE presupune exprimarea conceptelor cheie și a relațiilor într-o manieră formală sugerată de un limbaj de construire a sistemelor expert. De fapt, cognoticianul trebuie să dispună de cele mai potrivite limbaje de reprezentare a cunoașterii, necesare rezolvării problemei încă înaintea începerii formalizării. De exemplu, dacă problema se pretează la abordarea bazată pe reguli de producție se poate folosi ROSIE sau alt limbaj, ca limbaj de inginerie a cunoașterii, care poate exprima expertiza sub forma regulilor; dacă problema implică o abordare bazată pe cadre, se va folosi SRL sau alt limbaj care suportă lucrul cu cadre sau rețele.

Faza de IMPLEMENTARE presupune elaborarea programului de inteligență artificială cu un conținut, formă și integrare adecvate.

Conținutul se referă la domeniul problemei care trebuie elaborată explicit în timpul formalizării, sub aspectul structurilor de date, regulilor de inferență și strategiilor de control necesare în rezolvare. Forma este specificată în limbajul ales pentru dezvoltarea sistemului expert, iar integrarea presupune combinarea și reorganizarea variatelor piese de cunoaștere în vederea eliminării asamblării necorespunzătoare a structurilor de date, a regulilor și specificațiilor strategiilor de control și/căutare.

Implementarea trebuie să se desfășoare într-un ritm alert, întrucît una din rațiunile iumplementării prototipului inițial constă în controlul eficienței deciziilor de proiectare, care au fost luate în timpul fazelor precedente. Acest fapt ne arată că există o înaltă probabilitate a abandonării codului sursă inițial, din prima variantă a dezvoltării.

Faza de TESTARE implică evaluarea performanței și utilității prototipului de sistem expert și eventual revizuirea sa, dacă este necesar. Evaluarea prototipului este realizată de către expertul în domeniu, care ajută în mod corespunzător cognoticianul în toate operațiile unde este necesar. Sistemul expert prototip se va executa/testa pe cît mai multe probleme concrete, evaluîndu-se mereu utilitatea și performanțele. În cazul în care, evaluarea nu poate acoperi unele probleme sub aspectul conceptelor și relațiilor care s-au omis, ori a nivelurilor de detaliere necorespunzătoare, respectiv al mecanismelor de control greoaie, se va relua întreg ciclul de dezvoltare, de la prima fază, cu reformularea conceptelor, rafinarea regulilor de inferență și revizuirea întregului flux de control.

Evaluarea performanțelor sistemului prototip presupune observarea răspunsurilor la următoarele întrebări:

- Sistemul ia decizii cu care expertul este de acord?
- Regulile de inferență sînt corecte, consistente și complete?
- Strategia de control permite sistemului să trateze piesele de cunoaștere în ordinea naturală preferată de expert?
- Se dau explicații adecvate cu privire la acțiunile sistemului expert? Cum sînt îmbunătățite acțiunile?
-La testarea prototipului s-au acoperit toate cazurile posibile? Limitele probate sînt cele mai dificile?

Evaluarea utilității sistemului necesită, de asemenea, răspunsuri favorabile la o serie de întrebări și anume:

- Soluția obținută ajută utilizatorul într-un grad acceptabil?
- Viteza de răspuns la cereri satisface utilizatorul?
- Concluziile sistemului sînt organizate, ordonate și prezentate corespunzător nivelului adevărat de detaliere?

- Interfața de dialog cu utilizatorul este suficient de prietenoasă?

Sistemul expert prototip de cercetare trebuie rafinat și testat într-un laborator, înainte de a fi pus la testare în domeniul problemei. În perioada testării în domeniul real al utilizatorului pot să mai apară complicații, anomalii, care cer eforturi și un timp însemnat pentru corectare deoarece utilizatorii cer înaltă performanță, viteză mare, o bună fiabilitate, ușurință în exploatare și multă înțelegere atunci cînd ei greșesc. De aceea dezvoltarea sistemelor expert necesită un timp mare pentru testare înainte de a fi lansate în producție sau livrate în scop de comercializare.

3.4.3. Procesul de achiziție a cunoașterii

Achiziția cunoașterii necesare în vederea obținerii unui sistem expert puternic este una din primele obligații în dezvoltarea sistemelor expert. În prezent nu există metode informatizate performante pentru achiziția cunoașterii.

În procesul de achziție a cunoașterii intervin doi factori principali: expertul în domeniu și cognoticianul (inginerul de cunoștințe).

Cunoașterea necesară unui sistem expert poate fi procurată din surse multiple: cărți, rapoarte de cercetare, studii de caz, monografii, cercetare empirică și experiența personală. Neîndoielnic este faptul că sursa dominantă de cunoaștere este expertul în domeniu. În aceste condiții, cognoticianul va trebui să obțină cunoștințele printr-o interacțiune directă cu expertul (vezi fig. 3.27).

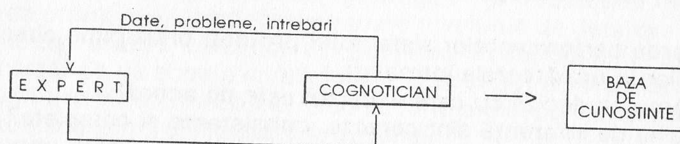


Fig.nr.3.27. Schema procesului de achiziție a cunoașterii

Interacțiunea constă într-o succesiune de interviuri intense, sistematice și prelungite pe parcursul a cîtorva luni.

În timpul unui interviu, cognoticianul prezintă expertului problemele de soluționat, ca și cum ar lucra într-un domeniu particular de interes (de exemplu, contabil, financiar etc.) și va obține regula sau metoda de rezolvare a fiecărui tip de problemă.

Intervievarea expertului trebuie să se desfășoare conform unor tehnici speciale:

- Observarea pasivă a expertului în timpul în care rezolvă o problemă din setul fixat în prealabil;

- Discutarea detaliată a fiecărei probleme pentru a vedea tipurile de date, cunoștințe și proceduri necesare;
- Descrierea problemei pentru fiecare categorie de răspunsuri;
- Se cere expertului să rezolve problema, dar și să o analizeze pentru cazuri reale, cu probarea raționamentului;
- Expertul oferă cognoticianului probleme de rezolvat cu noile concepte și formalisme introduse, ajutîndu-l să rafineze cunoașterea;
- Expertul rafinează și critică fiecare regulă notată de cognotician și evaluează împreună cu el strategiile de control folosite pentru selectarea regulilor;
- Cognoticianul prezintă cazurile soluționate de expert împreună cu prototipul demonstrativ și altor experți, în vederea comparării strategiilor și nepotrivirilor. În această manieră, sistemul expert final va fi rezultatul competenței mai multor experți.

În afara achiziției cunoașterii, la fel de importantă în dezvoltarea sistemului expert este alegerea corectă a scopului problemei și al limbajului pentru construirea sistemului. De fapt, acestea din urmă sînt două din cele mai dificile decizii de luat în dezvoltarea sistemelor expert. Din practică s-a observat că limbajele de inginerie a cunoașterii, cele mai actuale, dispun de noi posibilități de reprezentare a cunoașterii și s-a văzut că, pentru o anumită problemă, există numeroase limbaje care pot lucra la fel de bine.

Alegerea limbajului este necesară întrucît cognoticianul este familiarizat mai bine cu un anumit limbaj, un anumit limbaj poate fi mai performant și în funcție de calculatorul de care dispune utilizatorul, limbajul a fost realizat și testat nu și pentru aplicații din care face parte problema în cauză.

În toate cazurile se va urmări ca limbajul ales să ofere echipei de dezvoltare puterea și complexitatea de care are nevoie, să fie fiabil, ușor de întreținut și să dispună de toate funcțiile necesare pentru rezolvarea problemei sau aplicației.

* *
*

Dezvoltarea unui sistem expert necesită mijloace financiare, timp, echipament și personal, toate influențînd alegerea limbajului și instrumentului de dezvoltare și, mai ales, tipul (limbaj de programare LISP, PROLOG etc. sau limbaj de inginerie a cunoașterii). Limbajele de programare oferă mai multă flexibilitate, dar necesită dezvoltarea separată a bazei de cunoștințe și implementarea unui motor de inferențe pentru

accesarea bazei de cunoștințe. În acest mod, timpul este mai mare, iar rezultatele sînt mai apropiate de domeniul problemei. Limbajele de inginerie a cunoașterii oferă mai puțină flexibilitate, dar orientează mai bine procesul de dezvoltare și dispun de mecanisme de reprezentare și accesare a bazei de cunoștințe. În felul acesta, dezvoltarea sistemelor expert este mult ușurată și mai ieftină.

3.4.4. Dificultăți și limite cu privire la dezvoltarea sistemelor expert

Dezvoltarea unui sistem expert, în general, necesită mari eforturi financiare, de timp și energie intelectuală. Atunci cînd problema este potrivită și resursele financiare sînt angajate, investiția este rapid recuperată. În ceea ce privește resursele, este important de menționat mai întîi dificultatea de familiarizare a personalului din informatică cu înțelegerea și aplicarea tehnologiei sistemelor expert. Nu există cognoticieni experimentați și nici alte persoane capabile să construiască instrumente pentru dezvoltarea sistemelor expert; însăși instrumentele pentru construirea sistemelor expert înregistrează un progres impresionant, care trebuie urmărit.

În fiecare fază a dezvoltării sistemelor expert există dificultăți care trebuie înlăturate și anume:

I. Dificultăți generate de alegerea problemei potrivite.

a) Dacă efortul de dezvoltare este angajat într-o problemă dificilă, în care condițiile sînt restrictive, cognoticianul trebuie să dezvolte, mai întîi, un prototip de sistem mic, la care, după testare, să se procedeze la dezvoltarea completă. Și în acest caz, este necesar un mare efort de analiză, care va ajuta echipa de dezvoltare în eliminarea eventualelor probleme neadecvate dezvoltării sistemului expert. Se consideră probleme potrivite, pentru sistemele expert, diagnoza economică, planificarea, deciziile ș.a.. Nu sînt potrivite sistemele expert în reprezentarea cunoștințelor temporale și spațiale, pentru care se cere un consum mare de memorie.

b) Efortul de dezvoltare nu este întotdeauna motivat de dificultatea rezolvării problemei. După determinarea scopului problemei, se reevaluează eficiența în funcție de acest scop. În funcție de scop (de obicei eliminarea dificultăților întîmpinate de agentul economic într-un domeniu sau altul), echipa de dezvoltare se va decide asupra soluțiilor care vor reduce problema în mod semnificativ și va evalua fezabilitatea sistemului expert în noile condiții.

c) Problema este generală și complexă, determinînd un număr excesiv de reguli și obiecte ale bazei de cunoștințe. Această dificultate

determină obținerea de sisteme expert lente. Eliminarea acestei dificultăți necesită reexaminarea scopului problemei, a numărului de reguli, descompunerea pe subprobleme și construirea, mai întîi, a unui sistem prototip pentru o subproblemă.

II. Dificultăți datorate resurselor disponibile.

d) Timpul pentru dezvoltarea sistemului expert este foarte scurt și nu există personalul necesar dezvoltării. Se va decide angajarea celui mai bun personal specializat în dezvoltarea sistemelor expert, care va trece imediat la elaborarea unui plan adecvat de dezvoltare, cu estimarea timpului pentru fiecare fază.

e) Managementul crede că sistemul expert este un simplu program, care rezolvă o sarcină anumită și că orice programator poate dezvolta sisteme expert, dacă dispune de specificațiile de programare și un mediu de programare adecvat.

Sistemele expert se pot dezvolta numai de către un cognotician experimentat, singurul capabil să rafineze și să doteze cu modul explicativ astfel de sisteme.

III. Dificultăți din alegerea instrumentului de construire.

f) După ce alege un instrument de construire a sistemului expert, echipa de dezvoltare observă dificultăți de reprezentare a cunoașterii și structurilor de control din motorul de inferențe. Acest lucru impune un studiu amănunțit al instrumentului de construire a sistemului expert necesar pentru a fi ales, acela ale cărui caracteristici satisfac cel mai bine problema aleasă. După implementarea prototipului se știe mai bine dacă s-a făcut o alegere bună.

g) Alegerea celui mai bun instrument de dezvoltare se recomandă să fie făcută după consultarea mai multor cognoticieni , deoarece unul poate fi obișnuit cu un anumit instrument, pentru care are preferințe. A nu se decide construirea sistemului expert într-un limbaj de programare clasică, deoarece va crește timpul de dezvoltare din cauza lungimii neacceptabile a programului. Implementarea, testarea și rafinarea sistemului expert se recomandă să fie făcută într-un limbaj al inteligenței artificiale sau de ingineria cunoașterii, de înalt nivel, instrument cît mai specializat și apropiat de caracteristicile problemei. Viteza cea mai bună, în prezent, este asigurată de către instrumente cum sînt ROSIE, EMYCIN și AL/X ș.a.. Reimplementările costisitoare trebuie evitate. În orice caz, nu trebuie ales un instrument prea vechi și trebuie să i se controleze fiabilitatea, succesele în utilizare și cine-i este autorul. Cea mai practică modalitate de testare a sistemului expert este tratarea sa ca pe un student la examen: punerea problemei, observarea rezultatelor și capacității. Dacă descrierea problemei nu-i completă și cunoștințele în domeniu

insuficiente, nu se vor putea elabora modele de soluționare. Ca atare, se va urmări mai ales capacitatea de a emite diagnoze prin asociativitate în condițiile cunoașterii modelelor și metodologiilor pentru obținerea deducțiilor logice. Se va observa, totodată, flexibilitatea și adaptabilitatea, puterea de a genera informație inteligentă, cînd i se cere ș.a.m.d..

IV. Dificultăți datorate alegerii expertului în domeniu.

Se înțelege, că este întotdeauna nevoie de un expert cooperant și excelent instruit, de la care să se extragă regulile corecte în timpul dialogului cu cognoticianul. Competența în domeniu este aspectul cel mai important, alături de dăruirea în cooperarea cu specialistul în dezvoltarea sistemului expert. Să nu fie adversar al tehnologiei informatice. Se recomandă solicitarea unui angajament expertului înaintea începerii primei faze a dezvoltării. Să se stabilească și un expert suplimentar, eventual. Corectarea și modificarea regulilor sistemului expert să se facă cu sprijinul ambilor experți. În program se va utiliza obligatoriu terminologia folosită de experți în soluționarea problemei. Toți termenii utilizați se vor defini în prealabil, cu sprijinul experților.

Eventualul pericol ca regulile definite de expert să fie scurte și simple, față de situațiile cele mai complexe, poate fi înlăturat prin angajarea expertului în soluționarea unor probleme reale care se vor testa. Cu expertul se va face un dialog prietenos, în interviuri programate, egal distribuite față de durata dezvoltării sistemului expert. În felul acesta se vor evita tendințele de a termina mai repede ale expertului și se va obține creșterea contribuției sale la creșterea graduală a funcționalității sistemului expert. Se va pune accent pe interacțiunea cu experții în faza de dezvoltare a prototipului.

La testarea finală și rafinarea sistemului expert se recomandă folosirea altor experți în domeniu.

3.4.5. Viitorul sistemelor expert

Sistemele expert au înregistrat o extindere rapidă, guvernele și tot mai multe firme investesc masiv în această tehnologie. Se apreciază că agenții economici care vor ignora această tehnologie vor întîmpina mari dificultăți în competiția care îi așteaptă.

Un rezultat important al introducerii sistemelor expert îl constituie codificarea cunoașterii, în vederea tezaurizării sale în mari baze de cunoștințe, utilizabile oricînd în instruire și diseminarea informației, învățarea automată și, eventual, editarea unor cărți foarte bine documentate. Cunoștințele despre strategiile cele mai eficiente și procedeele lor de utilizare în domeniile de cunoaștere corespunzătoare se

vor folosi pe scară largă în viitoarele sisteme expert.

Cele mai practice echipamente pentru viitoarele sisteme expert sînt microcalculatoarele specializate, dedicate sistemelor expert. Se urmărește construirea de calculatoare cu pachete de programe expert încapsulate (integrate), care monitorizează funcționarea echipamentului.

Se înregistrează o creștere a interesului pentru noi domenii, în special economia, controlul proceselor, simularea sistemelor complexe, administrarea afacerilor, specializarea forței de muncă, contabilitatea, serviciile financiare, justiția, administrarea crizelor, alegerea alternativelor politice etc..

Sistemele expert vor conduce toate sistemele inteligente care deja se proiectează. Se menționează deja sisteme expert integrate, în care între sistemele expert propriu-zise și limbajele de ingineria cunoașterii în care-s construite, și echipamentul care le încapsulează nu-i nici-o distincție, deoarece formează o unitate perfectă. Sistemele expert integrate sînt încapsulate într-o singură piesă, sub forma unui complex electronic, care alcătuiește ceea ce se numește "sistem inteligent". Sistemele inteligente pot monitoriza și controla echipamentele de producție, asistă activitatea de reparații și întreținere, gestionează informațiile financiar-contabile aferente activității de producție ș.a.în deplină concepție integrată.

În cazul instalațiilor periculoase și al echipamentelor foarte scumpe, sistemele inteligente constituie soluția cea mai avantajoasă. Se prevede existența unor ierarhii de sisteme inteligente, care vor funcționa în rețea cu alte echipamente fizice, fie prin simpla comunicare cu echipamentele (fig. 3.37), fie prin comunicarea cu utilizatorul (fig. 3.38), așa după cum în prezent funcționează deja sisteme expert în conjuncție cu sisteme informatice clasice.

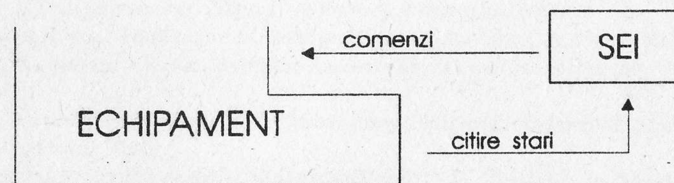


Fig. nr. 3.37. Sistemul expert integrat (SEI) în dialog cu un echipament.

Dezvoltarea sistemelor expert este un fenomen ireversibil, evoluția unei asemenea tehnologii va permite simularea celor mai complexe pro-

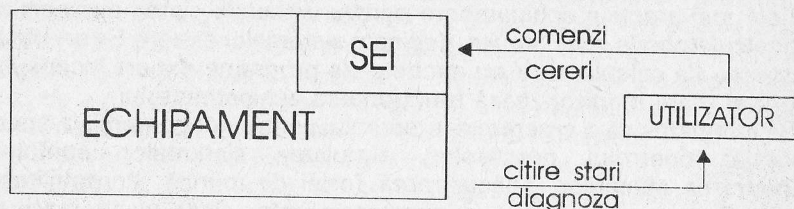


Fig. nr. 3.38. Utilizatorul în dialog cu sistemul expert integrat pentru controlul echipamentului

bleme ale cunoaşterii şi activităţii umane. Folosirea sistemelor expert are o specificitate foarte variată, ceea ce demonstrează generalizarea în toate domeniile. Aplicarea sa în domenii largi de activitate, obligă la elaborarea de sisteme multiexpert - cele mai proprii pentru abordarea multidisciplinară. Însăşi proiectarea sistemelor informaţionale şi informatice a devenit dependentă de proiectarea sistemelor expert. Iată, deci, o zonă de lucru viitoare cu adevărat copleşitoare pentru adevărata activitate intelectuală creativă.

BIBLIOGRAFIE

1. Abel, J. ş.a., Un prototype de système expert pour la finance, în "Journèe Systèmes Experts", 1984, Avignon, France;
2. Agapeyeff, A., British Computer Society Expert Systems, fifth generation, and UK suppliers, NCC Publications, Manchester, 1983;
3. Alzobaidie, A., Grimson, J.B., Experts systems and database systems: how can they serve each other?, în "Expert systems", 4(1), 1987;
4. Andone, I., Modernizarea contabilităţii întreprinderilor din ramura metalurgiei fierului, Teză de doctorat, Universitatea "Al.I. Cuza", Iaşi, 1984;
5. Barr, A., Feigenbaum, E.A., (Eds.), The Handbook of Artificial Intelligence, Los Alto, CA, William Kaufmann Inc., vol. I+II, 1986;
6. Blaga, L., Trilogia valorilor, Editura "Minerva", Bucureşti, 1987;
7. Bonczeck, R. ş.a., Foundation of Decision Support Systems, Academic Press, 1981;
8. Bonnet, A., Haton, J.P. ş.a., Système Experts: vers la matrise tèchnique, Inter-Editions, Paris,1988;
9. Bornes, A., Image et Intelligence Artificielle dans L'information Scientifique et Technique, cours INRIA, Paris, 1988;
10. Bostan, A., Sistem integrat financiar-contabil, în "Revista de finanţe, credit şi contabilitate", nr.2/1990;
11. Buchanan, B.G., Shortliffe, E.H., Rule-Based Expert Systems, Addison-Wesley,1984;
12. Călmăţuianu, V., Mihăiescu, P., Luchian, S., Sisteme expert şi utilizarea lor în medicină, Raport de cercetare, Univ."Al. I. Cuza", Iaşi, 1986;
13. Clocksin, W.F., Mellish, C.S., Programming in PROLOG, Springer-Verlag, New

York,1984;
14. Davies, R., (ed.) Intelligent Information Systems: progress and prospects, Chichester, Ellis Horwood Series in artificial intelligence, 1986;
15. Davis, R. Knowledge Based Systems in Artificial Intelligence, McGraw-Hill, New York, 1982;
16. Drăgănescu, M., Informatica şi societatea, Editura Politică, Bucureşti, 1987;
17. Eco, U., Tratat de semiotică generală, Ed. Ştiinţifică şi Enciclopedică, Bucureşti, 1982;
18. * * * "Enterprise et Comptabilité", Paris, nr.210, 1983;
19. Ericson, E.C., Ericson, L.T., Minolli, D., (eds.) Expert System Application in Integrated Network Management, Artech House Inc., 1989;
20. Farreny, H., Les systèmes experts: principes et exemples, Cepadues, 1985;
21. Farreny, H., Elèments d'itelligence artificielle, Hermes, 1987;
22. Feigenbaum, E., The Fifth Generation, Addison-Wesley Pub., Co., 1983;
23. Georgescu, I., Elemente de inteligenţă artificială, Editura Academiei Române, Bucureşti, 1985;
24. Goldsmidt, Y., Information for Management Decisions. A system for economic analysis and accounting procedures, Cornell University Press, Ythaca, 1970;
25. Goudran, M., Introduction aux Systemes Experts, Eyrolles Ed., 1984;
26. Guţu, Şt., Dumitraşcu, L., Elemente de inteligenţă artificială pentru conduccerea operativă a producţiei, Ed. Academiei Române, Bucureşti, 1983;
27. Harmon, P., King, D., Expert Systems, John Willey and Sons, New York, 1985;
28. Hayes-Roth, F., Waterman, D.A., Lenat, D.A., Building Expert Systems, Addison-Wesley, Reading, Massachusetts, 1983;
29. Hogger, J.C., Introduction to Logic Programming, Academic Press Inc., Orlando, Florida, 1984;
30. Hollingum, J., Expert Systems. Commercial Exploitation of Artificial Intelligence, IFS Ltd., Springer-Verlag, Berlin, New York, London, 1990;
31. Holloway, S., Bhabuta, L., Fourth Generation Languages and Application Generators, Gower Pub.Co., Ltd., Aldershot, 1986;
32. Holsaple, C-W., Whinshton, A-B., Manager's Guide to Expert Systems Using GURU, Dow Jones Irwin Homewood, Illinois, 1987;
33. Hunt, D.V., Artificial Intelligence & Expert Systems. Sourcebook, Chapman & Hall, New York, London, 1986;
34. Jackson, P., Introduction to Expert Systems, Addison-Wesley Pub.Co., Reading, Massachusetts, 1986;
35. Johnson, T., The Commercial Application of Expert Systems Technology, Ovum, London, 1984;
36. Keller, R., Expert Systems Technology, Prentice Hall Englewood Cliffs, New Jersey, 1985;
37. Kerschberg, L., (Ed.), Expert Database Systems, The Benjamin/ Cummings Pub. Co.,Inc., Redwood City, California, 1989;

38. Laurière, J.L., Representation et utilisation des connaissances, T.S.I., 1 et 2, 1982;
39. Levine, R.I., Drang, D.E., Edelson, B., A Comprehensive Guide to Artificial Intelligence and Expert Systems, McGraw-Hill Book Co., New York, 1986;
40. Liebowits, J., Desalvo, D.A., (eds.), Structuring Expert Systems: Domain, Design, and Development, Prentice Hall, Englewood Cliffs, New Jersey, 1989;
41. Maliţa, M., Maliţa, M., Bazele Inteligenţei artificiale, vol.I., Editura Tehnică, Bucureşti, 1987;
42. Marcus, C., PROLOG Programming, Addison-Wesley Pub. Co., Reading, Massachusetts, 1986;
43. Matsumoto, S., ES/SDEM-Software Development Engineering Methodology for Expert Systems, în "Future Generation Computer Systems", North-Holland, nr.5/1989;
44. McGraw, K.L., Knowledge Acquisition: Principles and Guidelines, Prentice Hall, Englewood Cliffs, New Jersey, 1989;
45. Meyer, B., Object-Oriented Software Construction, Prentice Hall, Englewood Cliffs, New Jersey, 1988;
46. Methie, L.B., Sprague, R.H., Knowledge Representation for Decision Support Systems, North-Holland, Amsterdam, 1985;
47. Michelsen, R., Michie, D., Expert Systems in Business, în "Datamation", nr.11/1983;
48. Minsky, M., A Framework for Representing Kowledge, în "The Psychology of Computer Vision", P.H.Winston (Ed.), McGraw-Hill Book Co., New York, 1975;
49. Mischkoff, H.C., Understanding Artificial Intelligence, Texas Instruments Information Pub. Center, Dallas, Texas, 1985;
50. Mullin, M., Object Oriented Program Design, Addison-Wesley, Pub.Co., Inc., Reading, Massachusetts, 1990;
51. Nilsson, N.J., Principles of Artificial Intelligence, Tioga Publishing, (Ed. I-a), 1980, Springer-Verlag (Ed. 2-a), 1982;
52. O'Shea, T., Eisenstadt, H. (Eds.), Artificial Intelligence Tools, Techniques and Applications, Harper & Row, 1984;
53. Pau, L.F., (Ed.), Artificial Intelligence in Economics and Management, North-Holland, Amsterdam, 1986;
54. Petriş, R., Bazele contabilităţii, Universitatea "Al.I.Cuza", Iaşi, vol. I+II, 1987;
55. Quinlan, J.R., (Ed.), Applications of expert Systems, vol.2, Turring Institute Press, Addison-Wesley Pub.Co., Sydney, Massachusetts, 1989;
56. Reitman, W., Artificial Intelligence Applications for Business, Ablex Pub.Co., 1984;
57. Rich, E., Intelligence Artificielle, Masson, Paris, 1987;
58. Roşca, I., Macovei, E., Davidescu, N., Zaharie, D., Răileanu, V., Proiectarea sistemelor informatice, A.S.E., Bucureşti, 1989;
59. Roy, R.H., Concepţii în management, Editura Tehnică, Bucureşti, 1983;
60. Rousset, M.C., TANGO: moteur d'infèrences pour une classe de systèmes experts avec variables, These de 3è Cycle, Orsay, 1983;
61. Rusu, D., Bazele contabilităţii, Editura Didactică şi Pedagogică, Bucureşti, 1980;

62. Schank, R.C., Abelson, R.P., Scripts, plans, goals, and understanding, Lawrence Erlbaum Associattes, 1977;
63. Scheer, A.W., Computer: A challenge for business administration, Springer-Verlag, Berlin, 1985;
64. Simon, H.A., Reason in Human Affairs, Stanford University Press, 1983;
65. Simon, H.A., Newell, A., Heuristic Problem Solving: the next advance in Operational Research, Applications of AI to Economics, Stanford University Press, 1984;
66. Şerbănaţi, L.D., Inteligenţa artificială, Ed. Tehnică, Bucureşti, 1985;
67. Thacker, I.R., Introduction to Modern Accounting, 2nd Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1974;
68. Turbuţ, Gh., (coord.), Inginerie de sistem, automatizări şi informatică în transporturi, vol.2, Ed. Tehnică, Bucureşti, 1989;
69. Turner, M., Expert Systems: A Management Guide, PA Computer and Telecomunications, London, 1985;
70. Voyer, R., Moteurs des Systèmes experts, Eyrolles, Paris, 1987;
71. Waterman, D.A., Lenat, D.B., (Eds.), Building Expert Systems, Addison-Wesley, Reading, Massachusetts, 1983;
72. Waterman, D.A., A Guide to Expert Systems, Addison-Wesley, Reading, Massachusetts, 1986;
73. Winston, P.H., Artificial Intelligence, 2nd Edition, Addison-Wesley, Reading, Massachusetts, 1984;
74. Winston, P.H., Inteligenţa artificială, Ed. Tehnică, Bucureşti, 1982;
75. Wiston, P.H., Horn, B.K.P., LISP, Addison-Wesley, Massachusetts, 1984;

* *
*

76. * * * Accounting Review, U.K., 1977-1991;
77. * * * Accounting & Business Research, U.S.A., 1977-1991;
78. * * * Accountability in Research, New York, 1989;
79. * * * Artificial Intelligence, North-Holland, 1989;
80. * * * Certified Accountant, U.K., 1988-1990;
81. * * * Decision Support Systems, North-Holland, '87-'89;
82. * * * Economia de piaţă. Instituţii şi mecanisme, în Supliment la "Tribuna economică", vol.I+II, 1990;
83. * * * Future Generation Computer Systems, 1986-1989;
84. * * * IEEE Expert, Intelligent systems & applications, Summer, 1987;
85. * * * Information and Software Technology, Butterworth, 1986-1991;
86. * * * Journal of Accounting & Economics, North-Holland, 1986-1990;
87. * * * Journal of Accountancy, USA, 1986-1990;
88. * * * Journal of Accounting Research, USA, 1988;

89. * * * Journal of Automated Reasoning, Kluwer Academic Pub., vol.5, no.1, 1989;
90. * * * Journal of Systems and Software, vol.11, no. 1, 1990;
91. * * * Knowledge-Based Systems, Butterworth, 1987-1991;
92. * * * Management Accounting, USA, 1985-1990;
93. * * * Metodologie cadru pentru elaborarea proiectelor directoare de informatizare şi a studiilor de fezabilitate, în "Tribuna economică", nr. 11/1991;
94. * * * Noul plan contabil general, în "Tribuna economică", nr. 11/1991;
95. * * * PC-Magazin, ADISAN, Bucureşti, nr. 1, 2, 3, 4/1990;
97. * * * Revista IF - calculatoare personale, Tg.Mureş, nr. 1, 2, 3, 4/1990, 1991.

SUMAR